LEARNING TO PERCEIVE AND PRODUCE WORDS IN CONNECTIONIST NETWORKS

Chan-Do Lee

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in Computer Science and Cognitive Science
Indiana University

September 1991

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral Committee

Michael Gasser, Ph.D.
(Principal Advisor)

Dennis Gannon, Ph.D.

Robert F. Port, Ph.D.

Gregory J.E. Rawlins, Ph.D.

July 1, 1991

DEDICATION

This thesis is dedicated to my parents, wife, and daughter.

Acknowledgements

I am most deeply indebted to my thesis advisor, Mike Gasser, for the inspiration and constructive criticism he gave to me over the course of my advanced graduate studies. He volunteered to be my advisor when I was almost ready to give up my graduate studies. Since then he has given generously of his time, knowledge, and constantly encouraged me. Even when he was away in California, he provided enough advice and stimulations to keep me running. I am also grateful to Bob Port for introducing me to speech research, helping me to launch my thesis research, and most importantly letting me use the facilities in the Phonetics Laboratory. I also want to express my thanks to the rest of my committee, Dennis Gannon and Gregory Rawlins, for thoughtful comments and ideas during the course of this work.

I extend my sincere thanks to Diane Kewley-port for introducing me to the arts of programming and encouraging me to study further. The group at the Phonetics Laboratory provided much help and support over the course of my graduate studies. I am grateful to Svën Anderson for enlightening discussions and much needed friendship. Jane Hardy and Cathy Rogers saved many of my days by rendering their help with my English.

Without the help of the office staff at the Computer Science Department, I would be still lost in the administrative maze. I am also grateful to the systems staff, too

many to mention, who were always helpful, even with the most insignificant requests.

Last but not the least, I thank the members of my family, for without continuous love and faith they entrusted in me, my studies and this thesis would not have come to fruition. I am sorry, Hannah, for not being able to closely watch you to grow. I will have more time for you. Thank you, mother and father, for incessant prayers and patience to wait for me for ten long years. I hope I can pay back what you have given me. To my wife Wea-Sook, I express my gratitude for her understanding, support, patience, and sacrifices throughout this rocky period.

Abstract

If there is one thing that linguists and laymen agree on, it is the central role of *words* in human languages. Yet there have been relatively few computational studies in understanding words themselves. How do we know how to say a word when we intend to utter something? What allows us understand the meaning of a word when we hear it? What is it that facilitates the association of a word with its meaning? This thesis describes how a connectionist network can learn to perceive and produce words.

Most traditional linguistic theories presuppose abstract underlying representations and a set of *rules* to obtain the surface realization. There are, however, a number of questions that can be raised regarding this approach: Where do underlying representations come from? How are rules formed and how are they related to each other? In this thesis, it is hypothesized that rules would emerge as the generalizations the network abstracts in the process of learning to associate forms (sequences of phonological segments comprising words) with meanings of the words and underlying representations could emerge as a pattern on the hidden layer.

Employing a simple recurrent network in which the hidden layer serves as a short-term memory that associates forms with meanings, a series of simulations on different types of morphological processes was run. The results of the simulations show that

Contents

List of Tables

List of Figures

1

Introduction

Being able to use natural languages[1] for the purpose of communication is one of the major characteristics that separate human beings from animals. The goal of achieving a computational account of perceiving and producing human language has been a major focus within Artificial Intelligence (AI) and cognitive science. Computers will not be able to help people perform many of the tasks they do until they can share the ability to use natural languages. Although a three-year-old child can speak and understand human language with relative ease, we have still not produced a computer program that can match this performance. Understanding language is hard. It requires not only language-specific knowledge, but also knowledge about the surrounding environment.

Spoken language is more basic than written language. Compared to speaking, employing written characters to communicate is a relatively new invention. But for computers, understanding written language is much easier than understanding spoken

[1]The term *natural language* refers to human language such as English and Korean, in contrast to *computer language* such as Pascal and C.

language, since the latter requires the former as a prerequisite plus other factors: digitizing the raw speech signal, smoothing the data, segmenting small chunks into larger chunks, labeling the segment, and analyzing the segment. This thesis addresses the problem of perception and production of words, a major part of language, from both spoken and written perspectives.

In this introductory chapter, I will discuss the problem, overview symbolic natural language processing (NLP) and connectionist NLP, draw the boundaries of the current research, and outline the contents of the thesis.

1.1 The Problem

In this thesis, I will pursue the problem of perceiving and producing words. A word is a minimal free form in language, that is, it is the smallest element that symbolizes and communicates a meaning, can occur in isolation and whose position with respect to neighboring elements is not entirely fixed. Even though it is the minimal free form, it is not the minimal meaningful unit of language, since it can often be broken into smaller units. For example, *pitchers* is a word since it can occur in isolation and can occur in different positions within the sentence. Nontheless it consists of three meaningful parts: *pitch*, *-er*, and *-s*. These minimal meaningful units are called *morphemes* (O'Grady et al., 1989). Studying words and their component morphemes is important and also very interesting, since the word is the central unit in language where the phonological and semantic components come together. Words are the dividing line between phonology/morphology and syntax/semantics/pragmatics.

There are a lot of research efforts directed to recognizing raw speech data in the

name of speech recognition. Much of NLP research has been dedicated to understanding written text. Yet there have not been enough studies in understanding words themselves. How do we know how to say a word when we intend to utter something? What makes us understand the meaning of a word when we hear it? What is it that facilitates associating a word with its meaning? There certainly is linguistic knowledge in our brains that makes these processes possible. Some involve phonology, some morphology, and some require an interaction between the two to explain the phenomena. We should be able to develop a computational device that can accommodate them. Without this middle ground our efforts to build a computer system that can understand human languages will be futile. We must develop a system that fills the gap between speech recognition and higher-level NLP in order to produce a reasonable speech understanding system. My thesis is a small, but hopefully a right step in this direction.

For my purposes, a word is not just a written piece of text such as *job*, or *top*, but a series of phonetic segments associated with a meaning, for example, the word with the meaning of *the highest point* is represented as

(1.1) /t/ TOP

(1.2) /ɑ/ TOP

(1.3) /p/ TOP

where the items in uppercase represent meaning and the expressions with phonetic characters surrounded by slashes refer to the word. My stimuli in the experiments were presegmented signals labeled with phonetic features. This was necessary to bridge the gap between two related, yet firmly divided fields of study: speech recognition with raw speech signals and high-level NLP with written text. The level of words in my study is higher than the raw speech signal, yet lower than the written

text, thus taking advantage of the ease of processing associated with written language understanding and not falling into the trap of ignoring real speech signals.[2] Even though I am not using real speech data, what I am dealing with here is spoken language; written language does not reflect the phonology that is one of my concerns. Nonetheless, the current study is also close to the domain of NLP that focuses on understanding written text, using lexical, syntactic, and semantic knowledge as well as required world knowledge. Thus what I am doing in this thesis is basically NLP, rather than precisely between NLP and speech.

In this thesis I concentrate on the issue of learning to perceive and produce words in terms of morphological processes. Morphological processes are generally regarded as symbolic, and rule-governed. Most traditional theories presuppose abstract underlying representations and a set of *rules* to obtain surface realizations. Modern generative grammar is based on the notion of "deriving" forms through the application of a series of rules, each of which takes a linguistic representation as input and yields one which is in some sense closer to the "surface". The idea is that behind surface forms are underlying representations, abstractions within which each morpheme has an invariant form. There are, however, a number of questions that have been raised regarding this approach: Where do underlying representations come from? How are rules formed and how are they related to each other? In this thesis, it was hypothesized that rules would emerge as the generalizations the network abstracts in the process of learning to associate forms (sequences of phonological segments comprising words) with meanings of the words and underlying representations would emerge as a pattern on the hidden layer. Through a series of experiments on different types of morphological processes, I will show how a recurrent connectionist network can prove this hypothesis.

[2]This point will be discussed in more detail in Chapter 2.

1.2 Symbolic and Connectionist Processing

A model based on a connectionist network is employed to achieve the goal of perception and production of words without the benefit of underlying representations and explicit rules. What, then, makes the two approaches to NLP, symbolic NLP and connectionist NLP, different? Why did I choose the connectionist model over the symbolic one? This section briefly overviews symbolic NLP, points out its weaknesses, describes connectionist NLP models, and emphasizes the advantages of connectionism for NLP.

The subject of study here is representational mental states and the principles of organization of computational systems. Over the last several years, there have been many discussions of the foundations of cognitive science. Is intelligence "the result of the manipulation of structured symbolic expressions"? Or is it "the result of the transmission of activation levels in large networks of densely interconnected simple units" (Pinker and Mehler, 1988, Introduction)?

While both classical symbolic and connectionist subsymbolic systems postulate mental representations, they differ in many ways. Fodor and Pylyshyn (1988) argue that only classical systems are committed to a "language of thought": that is, to representational states that have combinatorial syntactic and semantic structure. These systems are based on the productivity of thought, the systematicity of cognitive representation, the compositionality of representations, and the systematicity of inference. Cognitive capacities always exhibit certain symmetries, so that the ability to entertain a given thought implies the ability to entertain a thought with semantically related content. For example, from

$$P \wedge Q \rightarrow P,$$

we can infer

$$(A \vee B \vee C) \wedge (D \vee E \vee F) \rightarrow (A \vee B \vee C).$$

In a word, the basic building blocks of language are discrete symbolic entities that are manipulated by a set of rules.

Even though these seem to be good reasons for the symbolic postulate, there are several compelling deficits of rule interpretation when taken literally as a model of mind (Pollack, 1990):

1. *New rules are not easily learned in symbolic systems.* In symbolic systems all rules are hard wired into the system; hence making changes in the rules is difficult. In most cases, whenever a new rule is introduced, it has to be added explicitly to the code of the system.

2. *Symbolic systems do not easily scale up to real problems.* Most symbolic systems are for the toy problems with less than a thousand rules. It may require more than a thousand rules to solve real problems.

3. *Symbolic systems do not exhibit plausible behavioral profile.* Symbolic systems exhibit no temporal behavior that is psychologically plausible. Being able to accommodate temporal processing is very important, since language is inherently of a temporal nature. Language is not a static entity, but a dynamic one that must be processed continuously in real time. In symbolic systems this kind of temporal processing is very difficult.

4. *Symbolic systems are not generally capable of parallel processing.* Some decisions in human language processing are made by parallel constraint satisfaction. When a person hears an ambiguous word, he might use all the information

he has, lexical, syntactic, semantic, and contextual, to decide on its meaning. There is no easy way to create distributed memory and process according to parallel constraints in symbolic models.

5. *Symbolic systems do not show biological plausibility.* Symbolic systems have not been evolutionarily or neurally justified.

Connectionism can be viewed as a systematic attack on these problems, although any single connectionist model today only addresses one or two of them at a time.

1.2.1 Symbolic Natural Language Processing

Most computational approaches to NLP to date have come from the traditional symbolic perspective. In this view, the basic building blocks of the language and thought are discrete symbolic entities that are manipulated by a set of rules interpreted by a central control mechanism. This thesis is well manifested in the *physical symbol system hypothesis* (Newell and Simon, 1976) and the *language of thought hypothesis* (Fodor, 1976).

1.2.1.1 Examples of Symbolic NLP. Perhaps the prototype of symbolic models is the *logic-based system*, whereby a proposition is expressed discretely by a predicate and inference defines manipulations of predicate structures, in a manner that preserves truth values. A wide variety of alternative knowledge representations exist modeled after this prototype.[3]

[3]It is not the intention of this thesis to overview the past thirty odd years' collection of NLP research. Gazdar and Mellish (1989) gives a very good survey of symbolic NLP. Since symbolic language-understanding systems rely heavily on knowledge representation techniques, I will briefly introduce some of the techniques which were originally developed in the context of natural language research.

The logical formalism is appealing because it immediately suggests a powerful way of deriving new knowledge from old – mathematical deduction, while preserving combinatorial syntactic and semantic structure. In predicate logic, real-world facts are represented as *statements*, written as *well-formed formulas.*[4] A simple procedure is to use the basic rules of inference. A more powerful proof procedure is *resolution* which produces proofs by showing that the negation of the statement produces a contradiction with the known statements. For example, given the following sentences:

```
John likes all kinds of food.
Anything anyone eats and isn't killed by is food.
Bill eats peanuts and is still alive.
```

we can prove that *John likes peanuts* as shown in Figure 1.1. The logic-based system has been an indispensable tool for symbolic NLP for over thirty years.

Even though logical formalisms are very useful and easy to use, there often are cases when it is useful to collect properties of objects together to form a single description of a complex object. *Conceptual Dependency* (CD) (Schank, 1975; Schank and Abelson, 1977) provides a good example. CD is a theory of how to represent the meanings of sentences in a way that facilitates drawing inferences from the sentences and is independent of the language in which the sentence was originally stated. Its building blocks are a group of primitive actions, out of which higher-level actions corresponding to words can be built, and a set of allowable dependencies among the conceptualizations, which correspond to semantic relations among the underlying concepts.

[4] A well-formed formula (wff) is an expression that is formed according to predicate logic rules. For example, $P \wedge Q$ and $\forall_x man(x) \rightarrow mortal(x)$ are wff's, but $b\neg$ and $\exists_y mortal(y) \leftarrow man(y)$ are not.

Axioms in clause form:

1. ~food(x1) v likes(John, x1)
2. ~eats(y, x2) v killedby(y, x2) v food(x2)
3. eats(Bill, peanut)
4. ~killedby(Bill, peanut)

Prove: likes(John, peanut)

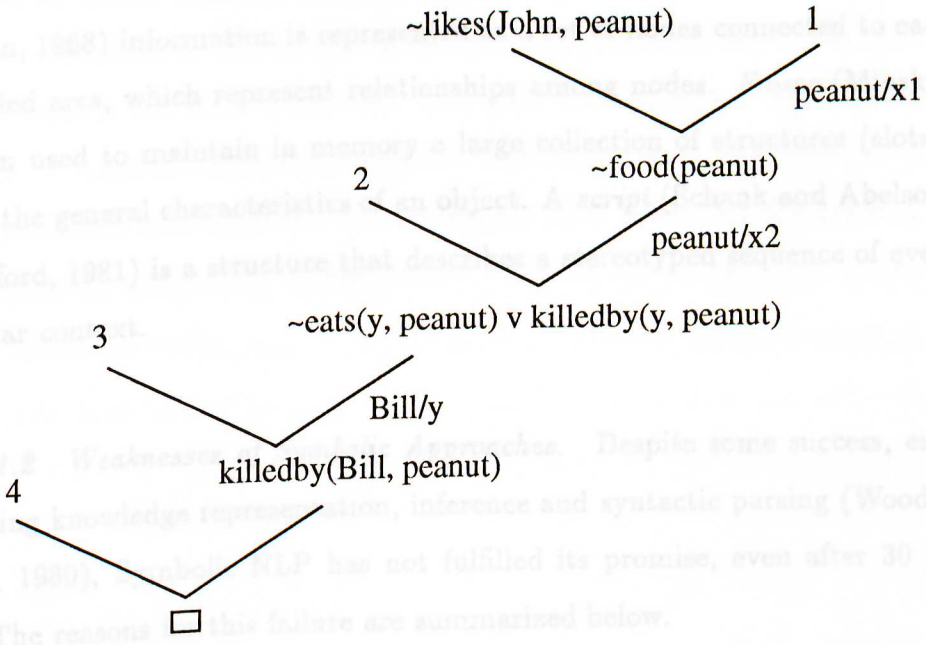

Figure 1.1: A resolution proof. In the figure, v denotes 'logical or', ~ 'not', and / 'substitution'.

An example of a simple conceptual dependency representation is shown below.

$$\text{John} \overset{p}{\Longleftrightarrow} \text{PROPEL} \overset{o}{\longleftarrow} \text{cart}$$

This CD represents the sentence *John pushed the cart.* In the figure, arrows indicate direction of dependency, the double arrow a two-way link between actor and action, *p* past tense, *PROPEL* a primitive act that implies the application of physical force to an object, and *o* object case relation.

There are other ways of representing structured knowledge. In a *semantic net* (Quillian, 1968) information is represented as a set of nodes connected to each other by labeled arcs, which represent relationships among nodes. *Frame* (Minsky, 1975) has been used to maintain in memory a large collection of structures (slots) representing the general characteristics of an object. A *script* (Schank and Abelson, 1977; Cullingford, 1981) is a structure that describes a stereotyped sequence of events in a particular context.

1.2.1.2 Weaknesses of Symbolic Approaches. Despite some success, especially concerning knowledge representation, inference and syntactic parsing (Woods, 1970; Marcus, 1980), Symbolic NLP has not fulfilled its promise, even after 30 years of study. The reasons for this failure are summarized below.

First, human language processing is extremely robust. Given enough context, listeners can understand novel well-formed words without much difficulty. Yet, symbolic models are notoriously brittle when presented with noisy or faulty input. There is no general mechanism in symbolic NLP that can deal with this problem.

Second, some decisions are made by parallel constraint satisfaction. When a

person hears an ambiguous word, he might use all the information he has, lexical, syntactic, semantic, and pragmatic (contextual), to decide the appropriate meaning of the word. It is difficult, if not impossible, to distribute memory and process in parallel in symbolic models. Most connectionist networks rely on distributed representations, where a large scale entity is represented by a pattern of activation over a set of units that themselves have conceptual representations composed of microfeatures.[5] But this is not easy in symbolic models where unitary symbols encode words.

Third, symbolic NLP does not give learning much attention. Most symbolic modelers follow Chomsky's (1965) thesis that a child is equipped with the biologically programmed *innate* capacity to acquire and utilize a linguistic system. The task of language acquisition is thus simply to fix upon the particular aspects of the linguistic system to which the child is exposed. The basic assumption is that language is not learnable. It requires a knowledge engineer who translates all the rules and surrounding environments into a system. Since rules are hard wired into the system, it is not environmentally adaptable. Whenever new rules are introduced, they must be wired into the system directly.

[5]This is not the only definition of "distributed representation". There are many coexisting definitions of this term. Two of the most common definitions are (1) the notion of simple extendedness just mentioned, i.e., using "many" units to represent a given item and (2) superimposition of representations (van Gelder, 1991). We have superimposition when there are multiple items being represented at the same time, but no way of pointing to the discrete part of the representation which is responsible for item A, the discrete part which is responsible for item B, and so forth. As shown in Chapter 5, the representation developed for a "plural" UR is a distributed one according to the "superimposition" definition .

1.2.2 Connectionist Natural Language Processing

Connectionism offers an alternative to symbolic processing. Even though it remains to be seen whether subsymbolic paradigm should replace the symbolic paradigm, at least connectionist representations show that the former succeeds in the area where the latter fails, especially "for studying how a cognitive system can possess knowledge which is fundamentally, *soft*, but at the same time, under ideal circumstances, admit good higher-level descriptions that are undeniably *hard*." (Smolensky, 1988, p.31)

1.2.2.1 Connectionism. Connectionism, or parallel distributed processing (Rumelhart and McClelland, 1986a; McClelland and Rumelhart, 1986), is an approach to cognitive modeling which assumes that knowledge is represented by weighted connections, spreading activations over large numbers of densely interconnected units.

A connectionist network consists of input units, which respond to stimuli from the outside world, and output units, which represent the system's response to that input. There may be one or more "hidden" units. Each unit has an activation value, which is updated by multiplying each incoming signal by the connection weight along which it is received, summing these inputs, and passing them through some function, thus obtaining a new value. A typical output function looks like the following:

$$O_{pj} = \frac{1}{1 + e^{-(\sum_i w_{ij} o_{pi} + \theta_j)}}$$

where O_{pj} is the jth element of the actual output pattern produced by the presentation of input pattern p, w_{ij} is the weight from the jth to the ith unit, o_{pi} is the output of ith unit for pattern p, and θ_j is a bias similar in function to a threshold. Processing involves activating input units; this activation spreads through the connections

to produce a pattern of activations on the output level. This is where the similarity among connectionist networks ends and differences come into play among many different network types and learning rules (see Hinton, 1989, for extensive overview of different learning procedures). A more detailed overview of connectionism will be given in Chapter 3.

1.2.2.2 Advantages of Connectionism for NLP. The weaknesses of the symbolic approach to NLP can be remedied by employing a connectionist approach. First, connectionist models are much more robust with regard to noise. Even with noisy or faulty inputs, the models can closely approximate the desired output, thus allowing the network to degrade gracefully. Content addressable memory and pattern completion arise naturally, making them ideal for filling in missing information.

Second, distributed connectionist networks use distributed representations rather than unitary symbols to encode words, allowing sometimes blending or blurring of the representations. Distributed representations allow the possibility of a fixed memory size. Adding a new rule does not require an increase of memory; all that needs to be done is to readjust connection weights.

Third, connectionist networks work as parallel constraint satisfaction; processing in connectionist models involves attempts to satisfy as many constraints as possible. This is what is required for language comprehension. Connectionist networks use massively parallel computation and have very powerful generalization capabilities.

Fourth, learning is fundamental to connectionist models. They can learn from exposures to examples, thus eliminating the need for the use of a priori rules. Also the network can learn to map one representation onto another. The network develops an internal representation on the hidden layer that approximates the relationship between two given representations.

Fifth, it can handle the temporal nature of language more easily. Language is not a static entity, but a dynamic one that must be processed sequentially in time. In symbolic processing temporal processing is very difficult, while any kind of sequential connectionist model can process temporal sequences, as will be shown in Chapter 3.

Sixth, the integration of representation and learning may provide a new theory. Connectionist networks are very good at extracting regularities, yielding new representations: it is the nature of the representation, uninfluenced by any theories, that holds the most promise for connectionist NLP. As most scholars would agree, one of the goals of studying NLP is to test a new theory and connectionism appears to point in the right direction for pursuing this rather ambitious goal.

1.2.3 Connectionism for Word Perception and Production

So far, symbolic NLP and connectionist NLP have been briefly overviewed and the differences highlighted. At this point, the question concerning both approaches and thus the problem at hand has to be answered. What made me choose connectionism over symbolic processing? Some might say, "Sure, connectionism may be a better way to handle lexical semantics, but why do you want to use it for word perception and production?" They might also suggest the feasibility of using a hybrid sort of approach (e.g. Kwasny and Faisal, 1991, use a hybrid system for syntactic parsing). But I am not seeking an engineering solution; this is meant to make a contribution to cognitive science. The subject of study is the **mind**, and the mind is not a steam engine, or a stored program computer. The key issue here is that of representation and learning; symbolic models do not tell us where underlying representations come from nor how rules are found and related to each other. In addition, symbolic models do not exhibit the temporal behavior that is fundamental in language processing.

1.3 Scope of the Project

In this thesis I report on a connectionist approach to the acquisition of morphological rules. There are several points to be made before going any further to make clear what can be expected from this project. I will try to draw the boundaries of my study by talking basically about what my model is *not*.

First, *my model is not a purely psychological model.* The overall objective of this study is for a connectionist network to learn to perceive and produce words. I am interested in how a connectionist approach can be used for both perceiving and producing words. But this study is not about language acquisition; I am not making any claims about children's acquisition of language skills, and my study is not concerned with developmental stages in language acquisition such as the U-curve, overregularization, etc. (Rumelhart and McClelland, 1986b; Chauvin, 1988; Marchman and Plunkett, 1989; Marcus et al., 1990). What, then, is my model trying to do? It might be possible for us to divide the types of models into "psychological models", "(computational) linguistic models", and "(purely) computational models". The last would be concerned only with particular categories of problems, independent of whether they have anything to do with human cognition. The second would be concerned with phenomena that are known to happen in natural language. The first would be concerned with modeling details of psychological processes. My model is mainly of the second type. It may also belong to some extent in the first, since one of the motivations of the current study is to shed some light on how a particular type of cognitive phenomenon can be accounted for without reference to explicit symbols or rules, though I obviously am not trying to model everything about this phenomenon.

Second, *the learning task is somewhat artificial.* As explained before, my primary concern in this study is words represented as phonetic segments, along with their

meanings. This falls into the category of NLP research, not into that of speech recognition, even though I am interested in the latter also. In the linguistic sense I will touch upon phonology and morphology. Input words were coded in such a way that they were presegmented and the segments were fed into the network one at a time. The network was given only the noiseless input data and a priori word boundaries. The model employed both meaning and form in a network to perform both the perception and the production task. The network was given both meaning and form from the beginning and was trained on both the production and the perception task from the onset. But it is quite possible that a person can hear words without perceiving their definite meanings at all; deriving the meaning from context and a knowledge of which aspects are relevant.

Third, *only fixed network architectures were used.* Even though one of my long-term goals is to have a network which evolves naturally by either dynamic node creation or some sort of skeletonization, in the project reported in this thesis only hand designed network architectures were considered. In particular, the number of hidden units was decided upon empirically, depending upon the tasks at hand. But note that it is not an uncommon practice among connectionists to design a network manually.

Fourth, *only one particular learning rule was used to train the network.* There are many different kinds of training procedures for connectionist models, among which the back-propagation learning algorithm is most commonly used. For reasons that will be explained in Chapter 3, I used the standard back-propagation rule to train the model.

1.4 Outline of the Thesis

This thesis describes how a connectionist network can learn to perceive and produce words. The next chapter raises the problem of words and computation. Related work is reviewed, similarities with the approaches that are taken in this thesis are pointed out, and drawbacks are singled out. Some theoretical background as manifested in the work of Langacker (1987) is summarized. The new approach called "performance grammar" employed in this thesis will be introduced. Hypotheses that motivated the current project are listed.

Chapter 3 explains connectionism in detail. It demonstrates connectionist learning procedures and describes how different connectionist models can achieve the task of learning temporal processes.

Chapter 4 describes the model in detail, including the network architecture used, and why this particular architecture was chosen.

Chapter 5 describes the experiments performed. It shows the stimuli used and explains how the system was trained. It provides detailed descriptions of the experiments: affixation, deletion, and mutation. Extensive analyses of the hidden layer are performed in an attempt to discover how the network encodes underlying representations on the hidden layer. The network is tested if it can exhibit knowledge transfer.

Chapter 6 discusses the results of the experiments. It examines the research hypotheses. It shows how the behavior of the model reflects some language universals: why does affixation task easier than deletion task? How does the network manage to generate affixes? What is it that makes the reversal rule so difficult for the network? The limitations of the current study are criticized and are related to directions for

future research.

Finally, Chapter 7 discusses the achievement made by this research and outline of future research problems suggested by this project.

2

The Problem

In this chapter, the problem of learning to perceive and produce words will be brought up, relevant literature will be reviewed, the approach taken in this project will be discussed, and hypotheses that motivated the current study will be listed.

2.1 Raising the Problem

Using language to communicate with others is one of the prominent characteristics that distinguish human beings from other animals. The goal of building a computer system that can exhibit the capability of understanding human language has been a long pursued AI project; yet we have not seen much success. What are the problems? It is time for us to go back to the drawing board and examine the causes of past failures and come up with new approaches to mend the old problems. The problem might lie in the lack of the cooperation needed to bring various theoretical subcomponents of language (such as syllable, morphemes, sentences and so on) together into a unified picture. Studying computational accounts of phonology and morphology has not

attracted many AI researchers, largely due to the fact that the apparent rule-like patterns exhibited by phonology and morphology did not appeal to the practitioners in traditional symbolic AI as an interesting problem. Since most work has been on English, which does not have much in the way of interesting morphology (unlike Finnish) most problems seem too easy to implement. The other reason might be the position of phonology and morphology as in between NLP and speech recognition. NLP researchers have used words as their tools and did not bother to go down a step further. Phonology is not relevant in written language, the focus of NLP. On the other hand, researchers in speech recognition have concentrated only on real low-level signals. Yet, it should be noted that an NLP system that deals with a language like Finnish, Russian, Turkish, Japanese, or Korean must pay attention to morphology; it is simply too expensive to store all the variant forms in the lexicon. It is always possible to have very language-specific hacks that know where to look for inflections[1] in a given language; but this is not interesting theoretically. As for phonology, if the language in addition involves complex morphophonemic processes at the boundaries between morphemes (as in Finnish, which has very extensive inflections), then the system also needs to handle these processes. Studying phonology and morphology is also interesting because of the parallel between syntax and sentence semantics and morphology and word semantics. Both involve compositionality and the problem of segmenting the input. But morphophonemics is perhaps simpler to study; for one thing, it seems not to involve recursion.

The unsuitability of approaches taken and tools used might have contributed to the cul-de-sac of NLP research. As discussed in Chapter 1, the traditional symbolic approaches to NLP lack many of the important features that are essential to a successful NLP system. In this thesis I study morphological processes with the

[1] *Inflection* is a morphological process that modifies a word's form in order to mark the grammatical subclass to which it belongs such as case, gender, number, tense, person, mood, or voice.

aid of a connectionist network, hoping to bring forward enough convincing results to bring some new insights into the NLP research community. In this section, I will show (1) what the problem is, (2) why it is important and interesting to study, and (3) why it is challenging.

2.1.1 Words and Computation

Among linguists of different schools of thoughts, one thing that is generally agreed upon is that a word is not an unanalyzable atom. A word is composed of smaller units and there is a hierarchy from more abstract units to less abstract units as shown in Figure 2.1. Of all the linguistic units shown in the figure, the word is the most familiar and it plays the major role in comprehending and generating spoken language. A word is a minimal free form. A very central part of comprehension is picking out the words in a stream of speech sounds, and a very central part of production is selecting lexical entries and grammatical features and then turning those abstract combinations into pronounceable forms. Studying words is important and also very interesting, since the word is the central unit in language where the phonological and semantic poles[2] come together. Words are the smallest unit in the hierarchy that embrace both meaning (semantic pole) and speech sounds (phonological pole) and can stand alone. Words are also the dividing line between low-level linguistic studies in phonology and morphology and higher-level studies in syntax, semantics, and pragmatics as can be easily seen from Figure 2.1.

[2]These concepts will be explained in Section 2.3.

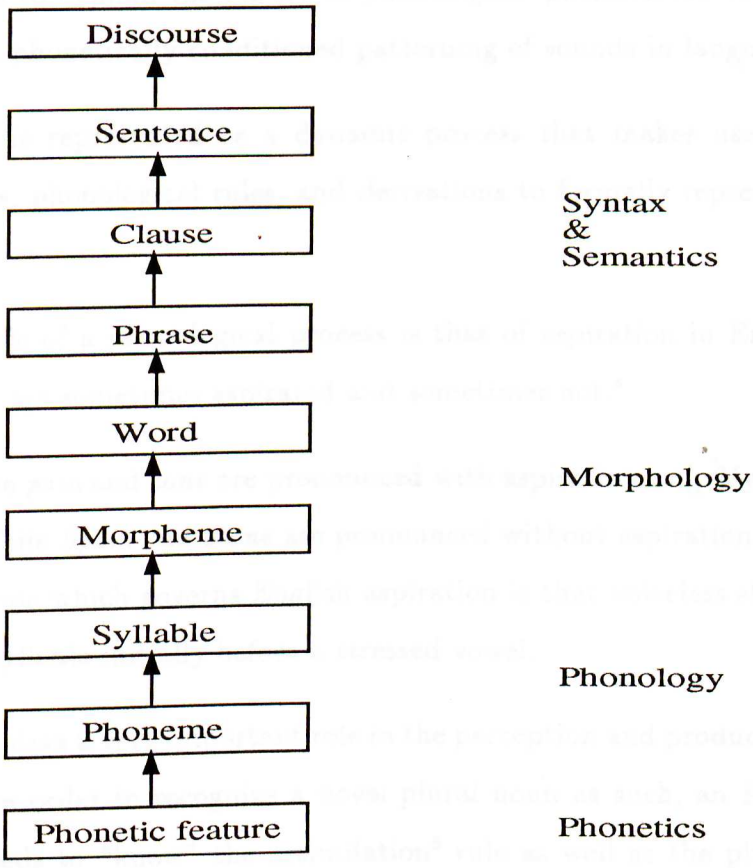

Figure 2.1: Hierarchical relationship among units that constitutes an utterance and components of linguistic study.

2.1.2 Phonology

Phonology is the study of the systems underlying the selection and use of sounds in the languages of the world. It focuses on the internal representation of sound units and tries to explain the nature of phonological phenomena. It deals with the sequential and phonetically conditioned patterning of sounds in language.

Phonology is represented as a dynamic process that makes use of underlying representations, phonological rules, and derivations to formally represent allophonic variation.[3]

One example of a phonological process is that of aspiration in English: English voiceless stops are sometimes aspirated and sometimes not.[4]

For example *pain* and *tone* are pronounced with aspiration as [p^{h}éyn] and [t^{h}ówn] respectively, while *Spain* and *stone* are pronounced without aspiration as [spéyn] and [stówn]. The rule which governs English aspiration is that voiceless stops in English are aspirated syllable-initially before a stressed vowel.

Phonology plays a very important role in the perception and production of a word. For example, in order to recognize a novel plural noun as such, an English listener presumably needs to "know" the assimilation[5] rule as well as the plural rule. The

[3]Predictable phonetic variants that are phonetically similar and in complementary distribution are called *allophones*.

[4]Stops are sounds made with a complete and momentary closure of airflow through the oral cavity. Examples of English stop consonants are as following:

span	[p]	stun	[t]	scar	[k]
ban	[b]	dot	[d]	gap	[g]
man	[m]	not	[n]	wing	[ŋ].

After the release of some voiceless stops in English, we can sometimes hear a brief delay before the next vowel. Since this delay is accompanied by the release of air, it is called *aspiration*.

[5]Assimilation is a phonological process that involves the modification of one or more features of

same applies to speech production. Without knowing the assimilation rule a speaker could not possibly add the right suffix to a novel noun, and would fail to produce the right plural form. This point will be brought up again in Section 2.4.

2.1.3 Morphology

Morphology is the study of word structures. How is it that we can use and understand effortlessly words that we have never encountered before? It is because many words consist of smaller components called *morphemes* which can be classified in a variety of ways and can be combined in different ways to create new words.

Even though the processes of word formation may vary from one language to another, all languages have the means to create new words and therefore exhibit the rule-governed creativity that is typical of human languages.

When we hear a reporter saying in a television news program that many of the homeless in American cities are former mental patients who were released because of a policy of *deinstitutionalization*, we understand quite effortlessly that it refers to the practices of releasing patients from hospitals for the mentally ill. We know this because we know what the word *institution* means, and we have unconscious command of English morphology.[6]

Once I had a friend from *Singapore*. While talking about him with another friend of mine, I found myself referring him as a *Singaporean*. Even though I did not know this word, I could easily make it up, since I have known that suffixing *-an* to a geographical name turns the word into a noun that means the inhabitant in the

a sound under the influence of neighboring elements (e.g., *in code* is often pronounced [ɪŋkʰówd]).

[6]This example is from O'Grady et al., 1989.

place, for example European from Europe.

2.1.4 Phonology and Morphology: Morphophonemics

The interaction between the phonological and the morphological components of grammar is reflected by the presence of allomorphs. One example of allomorphic variation is the pronunciation of the English plural suffix. The English regular plural morpheme has three variants: /s/, /z/, and /ɪz/, depending on the final segment in the noun stem. For example, the plural of *hat* is pronounced /hæts/ and that of *zone* is /zonz/, whereas *page* has the plural form /peʤɪz/. To account for these variants in traditional phonology, one must posit an underlying abstract representation, and one or more rules have to be invoked which transform the underlying representation into either /s/, /z/ or /ɪz/ given the correct environment. The derivation of a form like *hats* begins with the underlying representation of the morphemes *HAT* and *PLURAL*, and the rules turn these into the surface form /hæts/. As in the case of allophonic variation, the use of underlying representations and derivation by *morphophonemic* rule accounts for the morphophonemic alternations.

2.1.5 Morphological Constructions

All words consists of several smaller elements, that is, morphemes, and these can be classified in a variety of ways (e.g. **prefix** vs. **suffix**) and can be combined in different ways to create new words. Although the processes may differ, all languages have the means of creating new words and thus exhibit rule-governed behavior. Morphological processes can be classified according to two main criteria as the following:

1. Method of change

   Addition Affixation and reduplication are examples of addition found in human languages.

   Deletion Elision is the omission of an unstressed vowel or syllable to avoid certain phoneme clusters.

   Mutation Substitution is an example of mutation.

2. Place of change

   beginning (by addition: prefixation)

   middle (by addition: infixation)

   end (by addition: suffixation)

   suprasegmental[7] (tone changes; stress changes)

We can combine these two modes and make up many logically possible morphological rules, most of which I will list here. Where any natural language exhibits the kind of rules mentioned, I will give an example. Finally, I will explain which rules were chosen for the current study and why they were selected.

2.1.5.1 Affixation. Affixation is a type of simple word construction that involves an affix, a bound morpheme which occurs only in a particular position. Furthermore, each affix attaches only to a particular lexical category (either noun, verb, or adjective) and results in a word of another particular lexical category or inflection. We can further classify affixation into the following according to the place of affixation.

Prefixation A prefix is an affix that occurs in front of its stem.

Example. Sierra Popoluca (Mexico). (from Elson and Pickett, 1962, p.10)

[7] *Suprasegmental* refers to the intrinsic aspects of phones, such as pitch, loudness, and duration.

/kama/ 'cornfield' /i̲kama/ 'his cornfield' /aŋ̲kama/ 'my cornfield';
/way/ 'hair' /i̲way/ 'his hair' /aŋ̲way/ 'my hair'

Infixation An infix is an affix that occurs within another morpheme.
Example. Tagalog (The Philippines). (from O'Grady et al., 1989, p.96)

/takbuh/ 'run' /tumakbuh/ 'run + past';
/lakad/ 'walk' /lumakad/ 'walk + past'

Suffixation A suffix is an affix that occurs after its stem.
Example. (from Bergenholtz and Mugdan, 1979, p.59)

English. /hʌnt/ 'hunt' /hʌntɪd̲/ 'hunt + past';
German. /bɛt/ 'bed' /bɛtə̲n/ 'bed + plural';

Circumfixation A circumfix is an affix that surrounds its stem.
Example. (from Bergenholtz and Mugdan, 1979, p.59)

Tioko (North Togo). /bara/ 'woman' /m̲bara m̲/ 'woman + plural';
German. /frak/ 'ask (stem)' /g̲efrakt̲/ 'ask + perfect participle'

Transfixation A transfix is an affix that occurs side by side with its stem.
Example. Hebrew. (from Bergenholtz and Mugdan, 1979, p.59)

/ti̲fe̲l/ 'boy + singular' /tfa̲l/ 'boy + plural';
/kti̲e̲b/ 'book + singular' /ko̲tba̲/ 'book + plural'

2.1.5.2 Reduplication. A reduplicative affix is an affix that repeats all (full reduplication) or part (partial reduplication) of the stem to which it is attached.
Example. (from Bergenholtz and Mugdan, 1979, p.60)

Ilokano (Philippine). /talon/ 'field' /ta̲ltalon/ 'fields';
Indonesian /anak/ 'child' /anaka̲nak̲/ 'all sorts of children'

2.1.5.3 Deletion. Deletion is a process that removes a segment from the stem to avoid a certain phoneme cluster. We can further classify deletion process into the following classes.

Initial Deletion A segment is deleted from the beginning of a word. To my knowledge, there exists no human language which undergoes this type of rule as a morphological process.

Medial Deletion A segment is deleted from the middle of a word.

Example. Zoque (Mexico). (from Bergenholtz and Mugdan, 1979, p.98)

/nahp/ 'tread (stem)' /nahpa/ 'he treads';

/sihk/ 'laugh (stem)' /sikpa/ 'he laughs'

In order to account for the above data where the suffix /pa/ signals the third person singular, two rules are needed:

(1) /pp/ → /p/ and

(2) /XYZ/ → /YZ/, where X, Y, and Z are arbitrary consonants.

Rule (1) (*degemination*) changes */nahppa/[8] into /nahpa/; Rule (2) */sihkpa/ to /sikpa/.

Final Deletion A segment is deleted from the end of a word.

Example. French. (from Bergenholtz and Mugdan, 1979, p.99)

/kurt/ 'short (stem)' /kur/ 'short (masculine)';

/kurtə/ 'short + feminine suffix' /kurt/ 'short (feminine)'

We need the following two rules to account for the above data:

(1) /C/ → ∅ and

(2) /ə/ → ∅.

The first rule changes */kurt/ to /kur/, while the second rule changes */kurtə/ to /kurt/.

[8] In linguistic notation, the asterisk denotes ill-formedness.

2.1.5.4 Mutation. Mutation is a type of process where a segment is substituted by another segment. Mutation can occur:

in the middle (e.g. Yiddish. /kop/ 'head' – /kep/ 'heads'),

at the beginning (e.g. Mazahua (Mexico). /t'iʔi/ 'boy' – /c'iʔi/ 'boys'), and

at the end (e.g. English /bɪliːv/ 'believe' – /bɪliːf/ 'belief').

(from Bergenholtz and Mugdan, 1979, p.61)

Several different subclasses of mutation are possible, as shown below.

Ablaut Ablaut is the replacement of a vowel with a different vowel.

Example. English. (from Bergenholtz and Mugdan, 1979, p.61)

drink – drank – drunk.

Stress Shift Stress shift is a suprasegmental mutation which marks the difference between related grammatical categories.

Example. English. (from Bergenholtz and Mugdan, 1979, p.62)

/ɪm'pɔɹt/ 'import (verb)' – /'ɪmpɔɹt/ 'import (noun)'.

Tone Change Tone change is another type of suprasegmental mutation.

Example. Mongbandi (Congo). (from Bergenholtz and Mugdan, 1979, p.62)

/gwè/	'(he) went'	/gwé/	'(they) went';
/ŋgbò/	'(he) swam'	/ŋgbó/	'(they) swam'

Metathesis Metathesis is a change in the relative positioning of segments. Metathesis often results in a sequence of phones that is easier to articulate.

Example. Zoque. (from Bergenholtz and Mugdan, 1979, p.101)

/poj/	'run (stem)'	/popja/	'he runs';
/camcamnaju/	'enjoy (stem)'	/camcamnapja/	'he enjoys'

As shown earlier, the affix /pa/ signals the third person singular. Here the metathesis rule /jp/ → /pj/ changes */pojpa/ into /popja/ and */camcamnajpa/ to /camcamnapja/.

Reversal This is a special case of metathesis that reverses a whole word to construct a new word, which does not occur in any human language.

2.1.5.5 Complex Constructions. So far we have only seen simple constructions. It is quite possible to combine one or more simple construction types to yield a new word. I will list only three complex construction types among the very large number of possibilities.

Prefixation and Suffixation

Example. Tlingit (Alaska). (from Bergenholtz and Mugdan, 1979, p.73)

/ihɪtɪ/ 'your house' = /i/ '2. person' + /hɪt/ 'house' + /ɪ/ 'from'.

Suffixation and Mutation

Example. German. (from Bergenholtz and Mugdan, 1979, p.73)

/mɛnər/ 'men' = /man/ 'man' + /a→ɛ,ər/ 'plural'.

Pig Latin Pig Latin is a language game that combines suffixation and mutation where "the word-initial consonant of the English word is moved to the end and the vowel [e] is added after it. For example, the way one would say *Pig Latin* in Pig Latin would be [Igpe ætInle]." (Davis, 1991).

2.1.5.6 Summary. In this subsection, I survey possible word construction types. Most of these processes can be found in human languages. It would be difficult, if not impossible, to include all of the above possible cases in the current study. I selected a few that are typical construction types and are also easy to implement. For addition cases, only prefixation, infixation, and suffixation processes that involve a single segment were used in the experiments. Prefixes and suffixes were limited to assimilatory affixes. The infixation example involves the gemination process.

For the deletion cases, all three (initial, medial, and final) deletion rules were

tested. Like addition cases, only those that involve a single segment were included in the experiment.

There are many possible rules that can be classified as mutation rules. Among them tone change and reversal were used in the experiments. The reversal process was specially selected, because it does not occur in any human language, although some models are capable of simulating it.

Only one complex construction, Pig Latin, was considered in the current study.

2.1.6 The Challenge

In this thesis I concentrate on the issue of learning to perceive and produce words in terms of morphological processes. In some cases I am also looking at words which involve allomorphs. Morphophonemic processes involve both phonology and morphology.

Morphological processes are generally regarded as symbolic, and rule-governed. Accounting for such processes using connectionist networks presents many challenges that have been noted by researchers who subscribe to classical symbolic tenets, including Pinker and Prince (1988) and Fodor and Pylyshyn (1988). According to their arguments, connectionist models cannot command the compositional semantics that is supposedly essential to NLP. There have been a number of papers in connectionism refuting this argument (Elman, 1989a; van Gelder, 1989; van Gelder, 1990; Chalmers, 1990). My study is another attempt to show that connectionist networks are indeed capable of dealing with compositionality. The networks were taught to map combinations of meanings onto combinations of forms, and then decide which part goes with which. They had to discover how to map constituents of form onto constituents of

meaning and to use this knowledge to interpret and generate novel forms.

In classical symbolic systems the acquisition of underlying representations has been difficult to account for, and this fact has in part motivated the idea of innate predispositions for certain linguistic structures. The very existence of underlying representations that can be manipulated by rules was one of the points Pinker and Prince (1988) and Fodor and Pylyshyn (1988) made against the adequacy of connectionist systems in explaining morphological processes. If we can overcome these difficulties by explaining some morphological rules without the benefit of any *explicit* rules,[9] or underlying representations, it will strongly support the appropriateness of connectionist networks to the task of NLP.

Questions for a connectionist approach include the following: Are the underlying representations worth studying, those assumed by the symbolic account? Why are there certain morphological phenomena? And why are there not certain kinds of phenomena? If we assume that morphological phenomena are related to some form of cognitive processing, then the representations best suited to deal with these phenomena, whether they be symbolic or subsymbolic, might lead to an initial hypothesis as to what mental representations in general are like.

2.2 Related Work

Computational accounts of phonology and morphology were mostly ignored by researchers in cognitive science and artificial intelligence until the early 1980's, when Koskenniemi (1983) noticed the usefulness of finite state transducers and developed

[9]For the purpose of our discussion, a rule is a function which maps one representation onto another. By *without the benefit of explicit rules* I mean that in my system rules are not directly represented.

a two-level morphology model. His work influenced later research in computational morphology. Since 1988, when George Lakoff (1988a, 1988b) called the attention of the connectionist community to phonology as a challenging problem, there has been some fruitful research directed towards the problem of phonology. In this section a brief overview of some of related work both in phonology and morphology, be it symbolic or connectionist, will be presented.

2.2.1 Models of Past-tense Morphology

Rumelhart and McClelland (1986b) (hereafter referred to as "RM") showed that a simple two-layer pattern associator can acquire the marking of the past tense in English. Their model maps representations of present tense forms of English verbs onto their past tense versions, that is, it employs direct mapping:

$$Uninflected \rightarrow PatternAssociator \rightarrow Past,$$

without any involvement of semantic characterization. The RM model has been thoroughly scrutinized and criticized by Pinker and Prince (1988). The conclusion of their analysis includes the following criticisms:

1. *It easily models many kinds of rules that are not found in any human languages.* It is as easy to represent and learn a quintessentially nonlinguistic map, such as mirror-reversal of phonetic strings (e.g. *pit* to *tip*), in the RM model as it is to learn the identity map.

2. *It fails to capture central generalizations about English sound patterns. Elaboration tolerance* (McCarthy, 1988) is the ability of a model to be extended to additional problems. One example of this is the problem of *knowledge transfer*,

that is, how a pattern of activity learned by one network becomes interpretable by another. The RM model cannot be elaborated to explain the commonality between the /t/-/d/-/ɪd/ alternations found in regular past tense forms and the /s/-/z/-/ɪz/ alternations found in the third person singular, regular plural nouns, possessive and so on.

3. *It cannot handle the elementary problem of homophony.* Since distinct lexical items may share the same phonological composition, the notion of lexical representation distinguishes phonologically ambiguous words such as *wring* and *ring*. The RM model represents individual objects as sets of their features. Nothing, however, represents the fact that a collection of features corresponds to an existing individual.

Yet, contrary to what Pinker and Prince seem to argue, these weaknesses turn out to be specific to the RM model, rather than inherent to all connectionist accounts. Redefining the task to be one of perception and production results in constraints on the kind of rules that can be learned, permits homophonous words to be distinguished from one another, and makes possible generalizations across the sound system of the language.

Plunkett and Marchman (Marchman and Plunkett, 1989; Plunkett and Marchman, 1989; Plunkett and Marchman, 1990) modified the original RM model and answered some of the questions raised above by employing a three-layer network rather than the two-layer pattern associator used by RM. Each verb in their simulations consists of a Consonant-Vowel-Consonant (CVC) string, a CCV string, or a VCC string. Each string is phonologically well-formed, even though it may not correspond to an actual English word. Each vowel and consonant is represented by a set of phonological contrasts, such as voiced/unvoiced, front/center/back. This is

done with a set of 6 units by turning on the appropriate unit. After initial training on a subset of 20 verb stems, the vocabulary is gradually expanded until it reaches a size of 500 verbs. They report that the network undergoes reorganizations that result in a shift from a mode of rote learning to a systematic treatment of the verbs. Their findings are important because they give an indication of what is a *learnable* language and what is a *learnable* distribution of forms in the input. My approach in this thesis is similar in a way. That is, I investigate logically possible morphological processes to determine which are learnable and therefore likely to occur in real language.

However, neither Rumelhart and McClelland nor Plunkett and Marchman models actual language processing as it occurs in human beings: their models merely map representations of present tense forms of English verbs onto their past tense versions without any involvement of semantic characterization at all, whereas human beings accomplish the task of turning a form to meaning and meaning to form. In this thesis, I offer an alternative to the RM model which deals with two fundamental problems that are overlooked by classical accounts of morphological processes as well as the RM model: the relationship of production and perception and learning of underlying representations and rules. In doing so, I answer many of Pinker and Prince's criticisms.

2.2.2 Cognitive Phonology

Goldsmith (1991) departs from the psychological implausibility of current generative accounts of phonology and proposes phonology as a cognitive system, a system of contrasts and constructs that exhibits goal-directedness, which he calls *harmonic phonology*. This reflects some suggestions made in Goldsmith (1989) and Goldsmith (1990).

An intelligent system should modify its representations in such a way that the structures better satisfy conditions of maximal coherence and simplicity. A phonological description includes two things: a set of rules which describe the permitted (and unordered) transitions which a language permits, and a set of statements regarding well-formedness of various phonological structures, which Goldsmith calls phonotactics. The role of phonotactics is to interact with the rules so as to form better output. In such a system, a representation will always seek the position where it satisfies the phonotactics best. A representation of a given utterance U on a level L is a path from a starting point R_1 to a final resting point R_n. The final resting point R_n is that representation which is the best-formed of all points accessible to R_1 via the paths made available by the rules of the language on that level.

The model has three levels: the M-level (essentially the underlying, or morphophonemic, level), the W-level (the level at which pure syllable structure is established), and the P-level (phonetic level). Each level consists of the statement of tactics at its level, plus a set of intralevel rules. Hence, there are five classes of phonological rules: three sets of intra-level rules, that is, (M,M), (W,W), and (P,P), and two cross-level rule sets: (M,W), and (W,P).[10]

To summarize his hypothesis, Goldsmith proposes that the dynamic character of a phonological analysis be split into a number of subsections, corresponding to the individual linguistic levels, in such a way that we can identify the phonological dynamic in each case as an instance of maximally satisfying the constraints of the particular level in question. As a result, his theory does away with complicated rule ordering. As noted, this view parallels the basic tenets of connectionism. It is certainly within the realm of the conceivable that the types of generalizations that emerge from connectionist models may be closer to the sort that are needed in a new

[10]Since there is a hierarchy of levels in phonology, the (M,P) level rule is unnecessary.

model of phonology.

Goldsmith has made some concrete proposals along the lines of harmonic phonology for the treatment of stress in Goldsmith (to appear) and for the treatment of syllabification in Goldsmith and Larson (1990).

In both papers, he develops a formal model that allows for a simple and direct account of the facts within a specific language, and that is neither static nor derivational. His model does not have any hidden representations and computations happen locally as the results of establishing simple arithmetic relations for the activation of neighboring units. The network is trained using Formula 2.1 until it reaches an equilibrium.

$$x_i(t+1) = tanh(x_i(t) - \alpha x_{i+1}(t) - \beta x_{i-1}(t)) \tag{2.1}$$

where $x_i(t)$ is the activation level of the i^{th} element at time t, α is a parameter which affects the activation level of the left-hand neighbor, β affects the right-hand neighbor. This is a kind of lateral inhibition in which each element inhibits its two neighbors: the i^{th} element sends an inhibitory signal of strength αx_i to the preceding element, and it sends an inhibitory signal of strength βx_i to the element to the right. Each of these elements is a segment and is given a integer value between 0 and 8 following Goldsmith and Larson's sonority hierarchy (Goldsmith and Larson, 1990); when the elements under consideration are syllable, they are given the value 0 or 1 for the treatment of stress (Goldsmith, to appear). In Formula 2.1, the hyperbolic tangent (*tanh*) is needed to avoid the explosive nonconvergent properties of the formula inside the parentheses, since *tanh* is a function which smoothly squashes all values into the interval $(-1, 1)$.

He compares his model with traditional grammars as follows:

Architecture of the network	Universal grammar
Pattern of connection weights	Language-specific grammar
Activation vector	Representation of a specific expression

He reports that his model derives the correct results for stress rules and syllabification in certain languages, though with a good deal less machinery than most connectionist networks, and no intermediate hidden representations. The forces at work affect each other simultaneously, and seek a stable resolution of their requirements.

Lakoff's cognitive phonology (Lakoff, 1988a; Lakoff, 1988b), a development of the ideas of Goldsmith described above, is another effort to eliminate the need for rule orderings. Ordered-rule interaction is a necessary foundation for conventional generative phonology. For example, consider the pronunciation of *writing* /ˈɹaɪtɪŋ/ and *riding* /ˈɹaɪdɪŋ/. In some dialects of English, these two words differ phonetically only in that the stressed vowel of the former is shorter than the stressed vowel of the latter, the medial consonant in both cases being a voiced flap /ɾ/. To account for this phenomenon, two rules are applied in the order specified below:

1. *Vowel Lengthening.* Make vowel long before voiced consonant.
2. *Flapping.* Change /t/ or /d/ to flap /ɾ/ between vowel, if the first vowel is stressed.

If we apply Rule 2 before Rule 1, we have the same incorrect pronunciation /ˈɹaɪːɾɪŋ/ for both *writing* and *riding*, as shown in Table 2.1.

Cognitive phonology, as Lakoff termed his theory, is free of the rule ordering constraints that make generative rules computationally awkward. It uses a multilevel

Table 2.1: Derivations of the pronunciation of *writing* and *riding*. The left half shows a wrong derivation due to the incorrect rule ordering. The right half shows the correct derivation. In the table, UR denotes the underlying representation, Flap the Flapping rule, VL the Vowel Lengthening rule, and PR the surface phonetic representation.

wrong	"writing"	"riding"	**correct**	"writing"	"riding"
UR	/ˈɹaɪtɪŋ/	/ˈɹaɪdɪŋ/	UR	/ˈɹaɪtɪŋ/	/ˈɹaɪdɪŋ/
Flap	/ˈɹaɪɾɪŋ/	/ˈɹaɪɾɪŋ/	VL	—	/ˈɹaɪːdɪŋ/
VL	/ˈɹaɪːɾɪŋ/	/ˈɹaɪːɾɪŋ/	Flap	/ˈɹaɪɾɪŋ/	/ˈɹaɪːɾɪŋ/
PR	[ˈɹaɪːɾɪŋ]	[ˈɹaɪːɾɪŋ]	PR	[ˈɹaɪɾɪŋ]	[ˈɹaɪːɾɪŋ]

representation for the utterance, to which multiple rules may apply in parallel. The M (morphophonemic) level represents the underlying form of an utterance, the P (phonemic) level is an intermediate level, and the F (phonetic) level is the derived surface form. Note that Lakoff's three levels are basically the same as Goldsmith's, but with different names. Phonological processes expressed as rules in standard theories are instead treated as "constructions" of two types:

1. Cross-level M-P and P-F constructions, which state the allowable correlations between levels.
2. Intra-level P and F constructions, which state the well-formedness constraints within a level.

There are no explicit rule orderings; all constructions at a given level apply simultaneously. However, by allowing both inter- and intra-level constructions, the theory

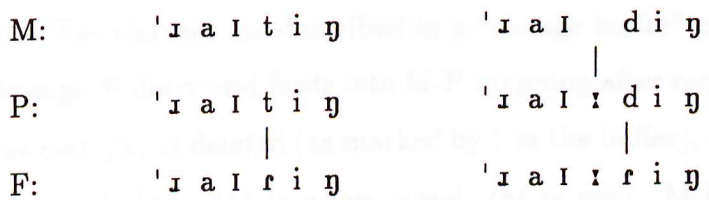

Figure 2.2: Cognitive phonology derivations of *writing* and *riding*. The lines indicate sanctioned correspondences. The rest is identical and is sanctioned by default associations.

achieves the effect of extrinsic rule orderings. Figure 2.2 shows the cognitive phonology derivations of the words *writing* and *riding*. The effect of the P-level constraint can be seen in the middle line of the figure: the penultimate vowel has been lengthened for /ɹaɪdiŋ/ since *d* is voiced, but no change has been made to /ɹaɪtiŋ/. A second mapping takes P-level representations to F-level representations. At F-level we see that both /t/ and /d/ has been changed to flap /ɾ/ since the first syllable /ɹaɪ/ is stressed.

Even though cognitive phonology is founded on connectionist principles, Lakoff does not specify how it should be modeled. Touretzky and Wheeler (Touretzky, 1989; Touretzky and Wheeler, 1990a; Touretzky and Wheeler, 1990b; Touretzky and Wheeler, 1990c; Touretzky et al., 1990; Wheeler and Touretzky, 1989; Wheeler and Touretzky, 1990) have developed a connectionist implementation of Lakoff's ideas, making some modifications along the way. This implementation utilizes a "many maps" architecture to manipulate sequences of phonemes at multiple levels. Basically this model, M^3P as they call it, uses many maps, such as M-P map and P-F map to account for inter-level mappings. It uses a higher level buffer and inter-level constructions to accompany the mapping matrix and this matrix in turn produces the correct lower level representations. An example is shown in Figure 2.3. In this example, the derivation involves three changes to the string: a mutation, a deletion,

and an insertion. The changes are described in a "change buffer" called P-deriv. The M-level goes through P-deriv and feeds into M-P mapping after required changes: /i/ is inserted at the end, /k/ is deleted (as marked by 1 in the buffer), and /a/ is changed into /e/ (as indicated *-low.* /a/ is a low vowel, /e/ is not). M-P mapping matrix derives the phonemic representation, right-justified in the P-level output buffer, in one parallel step. The mapping matrix makes sure that there are no gaps or collisions in the output caused by simultaneous insertions and deletions.

To explain intra-level constraints such as vowel harmony, Touretzky and Wheeler use a clustering mechanism. It is controlled by a small number of language-specific parameters. They include the type of the cluster, the trigger which lists the preconditions of the rule and the elements which can undergo a change, which is the last of the parameters to be specified. The clustering mechanism is implemented by an additional mapping matrix. Figure 2.4 shows an example of a clustering rule. In the example, round vowels are triggers and the 'element' bit is turned on if a vowel agrees with the trigger in its specification for the feature [high]. Sequences of vowels in which the 'element' bit is activated form a cluster and are changed into round vowels. In a hypothetical example with four vowels, /uCiCiCa/ becomes /uCuCuCa/ where C denotes any consonant. Here /u/ is a trigger and subsequent two /i/'s are elements, since both /u/ and /i/ are high vowels.

Despite of its powerful mechanisms that can deal with complex rule interactions, Touretzky and Wheeler's model still shares several points with standard models: (1) the assumption of abstract levels and underlying forms, (2) the presence of hard wired rules, (3) and the nature of the model as a *competence* model.[11]

[11]They are aware of that their program is not a realistic model of human language acquisition.

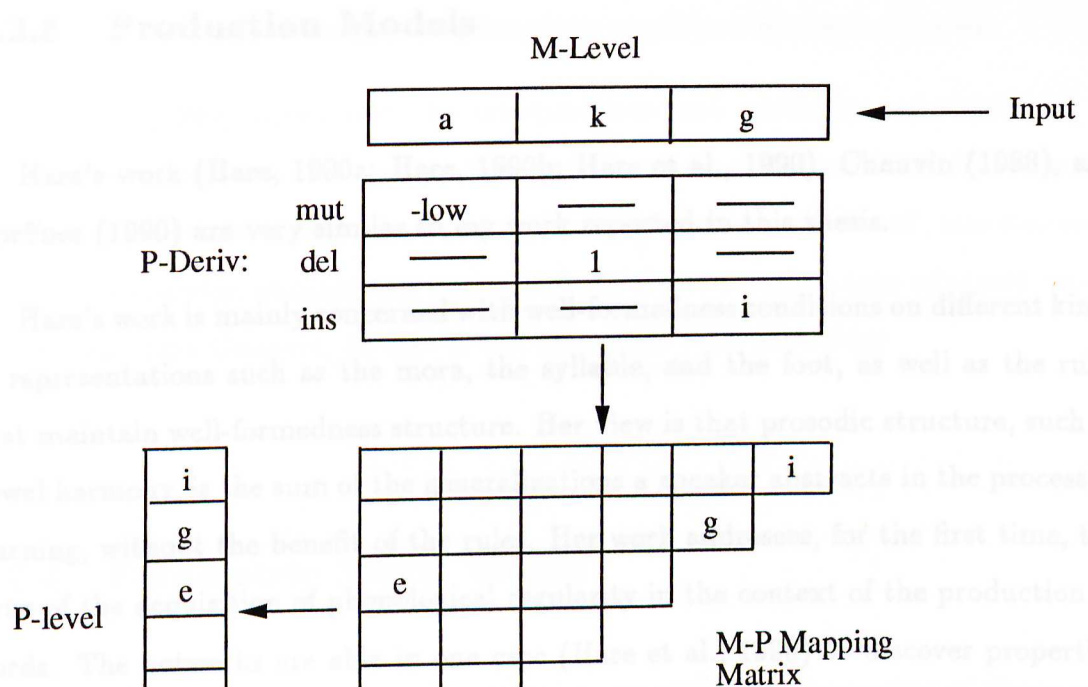

Figure 2.3: Example of M³P's M-P map. This figure shows how an M level representation of an utterance, /akg/, is mapped to a P level representation, /egi/. This figure was adapted from Touretzky and Wheeler, 1990b.

Yawelmani vowel harmony – P-F mapping:

Cluster type:	[+syllabic]
Trigger:	[+round, αhigh]
Element:	[αhigh]
Change:	[+round]

Figure 2.4: An example of a clustering rule. The trigger of a cluster is a round vowel of a given height, and the elements are the subsequent adjacent vowels of mapping heights (α *notation* is used to mark the same pole for a feature. For example, an α may be read as '+'. If it is, all other αs in the same rule are to be read as '+'.). Application of the rule causes elements to become round. This figure is taken from Wheeler and Touretzky, 1989.

2.2.3 Production Models

Hare's work (Hare, 1990a; Hare, 1990b; Hare et al., 1990), Chauvin (1988), and Dorffner (1990) are very similar to my work reported in this thesis.

Hare's work is mainly concerned with well-formedness conditions on different kinds of representations such as the mora, the syllable, and the foot, as well as the rules that maintain well-formedness structure. Her view is that prosodic structure, such as vowel harmony, is the sum of the generalizations a speaker abstracts in the process of learning, without the benefit of the rules. Her work addresses, for the first time, the issue of the acquisition of phonological regularity in the context of the production of words. The networks are able in one case (Hare et al., 1990) to discover properties of Turkish syllable structure and in the other (Hare, 1990a; Hare, 1990b) to discover detailed features of Hungarian vowel harmony. My work is similar to Hare's work, but it differs in two ways. First, I investigate the power of a somewhat different type of network. Second, I am concerned with both production and perception as achieved by a single network that can accommodate both processes.

Chauvin (1988) examines empirical results related to first-word acquisition in infants and explores how and why similar phenomena occur in a PDP model. During learning, a neural network is exposed to a micro-world composed of categories made of clusters of "images" and of labels attached to these clusters. The basic architecture of the network allows encoding of labels and images in a common level of representation and subsequent extraction of labels from images and images from labels. This task of comprehension and production applied to a neural network is very similar to my approach presented in this thesis. The difference lies in the representation of linguistic forms. Chauvin concentrates on semantics ignoring phonology, and so does not need a dynamic representation of words, while the particular problem domain I have been

exploring, that is, morphology and phonology, requires a dynamic network.

Dorffner (1990) shows how the interpretation and generation of words can be modeled in a connectionist model. His model has an architecture that learns words by categorizing sensory input from two channels, 'phonetic' and 'visual', into concepts, and then associates co-occurring concepts from the two realms into arbitrary words or symbols. Like Chauvin, he also is not concerned with phonology.

2.2.4 Parametric Stress Model

So far I have reviewed some related work in connectionist processing of phonology and morphology. There is more work done in symbolic paradigm, among them I will cite just two prominent works.

Chomsky (1965) argues that a child is equipped with biologically programmed innate capacity to acquire and utilize a linguistic system. The task of language acquisition is thus simply to finetune the particular aspects of the linguistic system to which the child is exposed. The nature of the problem can be represented by the diagram illustrated in Figure 2.5. Based on this theory, Dresher and Kaye (1990) constructed a parameterized, symbolic model for the acquisition of stress systems. This model assumes the existence of 11 preset parameters as a part of universal grammar; the task of the model is to fix the value of each parameter to come up with LT. So much given information[12] makes us wonder whether a child really learns a stress system this way.

[12]Obviously, they do not think they have started with *too* much information.

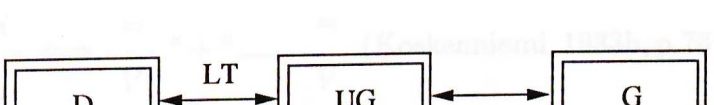

Figure 2.5: The Projection Problem. Given D, the data of language L, and a grammar, G, which is the grammar of L, G is acquired via UG, a set of innate and universal cognitive principles. If UG is construed as a set of rather abstract universal principles, then a learning theory, LT, relates D to UG.

2.2.5 Two-level Morphology

Koskenniemi (1983a, 1983b, 1984) puts forward a morphology in which all rules apply simultaneously, and in which each rule can be compiled into a finite state transducer (FST). An FST is a special kind of finite state automaton that inspects two (or more) symbols at a time and proceeds accordingly.[13] His theory posits a "two-level" model for morphological analysis (recognition) and synthesis (production). The model differs from generative morphology in that it proposes parallel rules instead of successive ones, thus avoiding complicated and often troublesome rule interactions. The name "two-level morphology" reflects the setup where only the underlying lexical and the surface levels ever "exist". All two-level rules express correspondences between lexical and surface forms and involve a surrounding context. The basic structure of two-level rules is:

$$\text{CP} \quad \text{OP} \quad \text{LC} \_\_ \text{RC}$$

Here CP refers to a lexical/surface correspondence, LC and RC refer to the left and

[13]Regular finite state automata inspect only *one* symbol at a time.

right context, respectively, and OP an operator indicating what kind of a relation is stated between the pair in the LC and RC. An example of a rule is

$$i \underset{j}{} \Longleftrightarrow \underset{V}{=} \text{“}+\text{”}\_\_ \underset{V}{=} \quad \text{(Koskenniemi, 1983b, p.78).}$$

This says that a plural /i/ is realized as [j] between elements that are realized as vowels (the '=' means, roughly speaking, 'don't care', and the '+'[14] has the effect, in this context, of flagging plurality). Such a rule gets compiled into the automaton shown as a matrix in Table 2.2. Here the rows correspond to the state and the columns to the symbols (for the particular input alphabet of the automaton). Final states have a colon after the state number whereas nonfinal states have a period. This FST can be diagrammed as in Figure 2.6. Below is a demonstration of the procession of the automaton in a configuration:

Input:	t	a	l	o	&	+	i	e	n	
	t	a	l	o	0	0	j	e	n	
State:	1	1	2	1	2	2	3	4	2	1

The other alternative, *taloien*, would have failed, because the transition on column 1 in state 5 is zero.

The Koskenniemi-style approach is described and evaluated in Gazdar (1985). Koskenniemi's work directly influenced the implementation of KIMMO system by Karttunen and his students (Karttunen, 1983). Finite-state morphology or phonology analyses exist for many languages, including English (Karttunen and Wittenburg, 1983) and Semitic (Kataja and Koskenniemi, 1988). Barton, Berwick, and Ristad (1987) have shown the method to be NP-hard, to which Koskenniemi and

[14] + is a boundary in front of case endings, one of eight morpheme boundary symbols Koskenniemi uses.

Table 2.2: An example of FST in a tabular form for a plural rule in Finnish. From Koskenniemi, 1983a, p.96.

	=	+	i	i	X	=
	V	0	=	j	0	=
1:	2	1	2	0	1	1
2:	2	3	2	0	2	1
3:	2	3	5	4	3	1
4.	2	4	2	0	4	0
5:	0	5	0	0	5	1

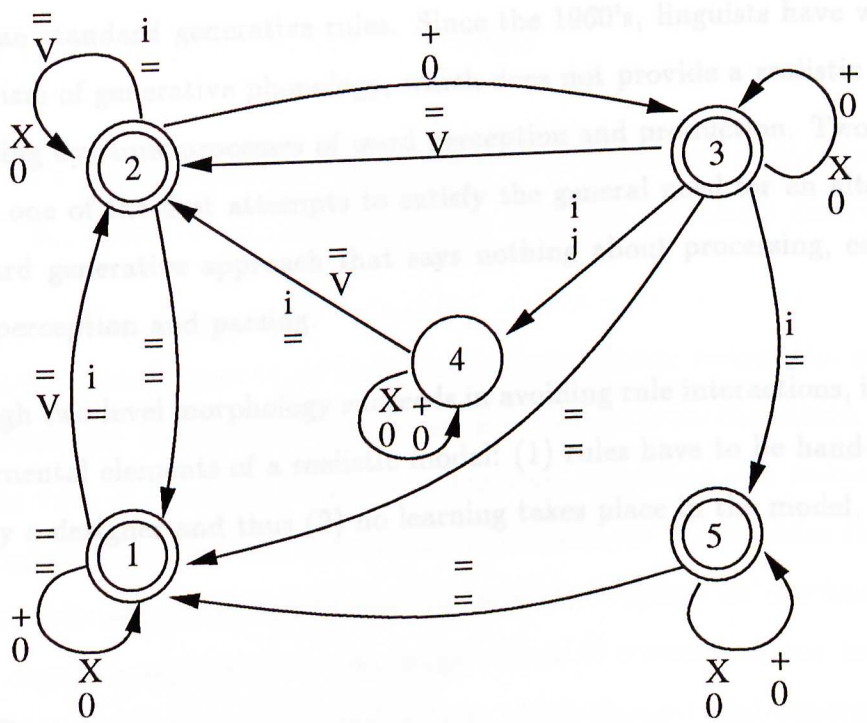

Figure 2.6: An example of FST for a plural rule in Finnish. From Koskenniemi, 1983a, p.96.

Church (1988) reply that theoretically Barton et al.'s analysis is correct, but in natural languages there are many constraints which block computational explosion. There are further theoretical developments that try to mend some of the problems posed by Koskenniemi's original approach; Bear (1988) explains how phonological rules may be employed instead of transition tables that are cumbersome to develop and refine; Carson (1988) shows that both unification and FST can be applied to phonological parsing.

Two-level morphology avoids rule interactions, since rules (the automata) work together in parallel; a configuration is accepted if all rules pass. One contradicting rule is enough to ruin the correspondence. The main idea is that rules can make explicit reference to levels in their environments. This makes them more powerful in a sense than standard generative rules. Since the 1960's, linguists have widely used the formalism of generative phonology, which does not provide a realistic framework for describing dynamic processes of word perception and production. Two-level morphology is one of the first attempts to satisfy the general need for an alternative to the standard generative approach that says nothing about processing, especially in regard to perception and parsing.

Although two-level morphology succeeds in avoiding rule interactions, it still lacks two fundamental elements of a realistic model: (1) rules have to be hand-coded and supplied by a designer and thus (2) no learning takes place in the model.

2.2.6 Summary

So far, some of the models that deal with computational phonology and morphology have been surveyed. Some of the models lack learning, while others are not

psychologically plausible.

My model is an attempt to overcome many of the problems raised by the models mentioned above by having both segmental and semantic inputs and outputs in the network, thus performing the more psychologically plausible process of the production of a sequence of segments given a meaning or the selection of a meaning given a sequence of segments. My model is a *dynamic* model. The model is given one segment at a time as input. Furthermore, my model does not presuppose any linguistic parameters, abstract underlying representations, or rules. These points will be made clear in the next few chapters.

2.3 Theoretical Background

Langacker's *cognitive grammar* (Langacker, 1987) provides a useful terminological framework. His theory is compatible with the connectionist model described in this thesis.

His grammar posits just three basic types of structures: **semantic**, **phonological**, and **symbolic**. Symbolic structures are not distinct from the others, but rather combine the two. A symbolic structure is **bipolar**, consisting of a **semantic pole**, a **phonological pole**, and the association between them. He assumes that one can validly postulate **semantic space** and **phonological space** as two broad aspects of human cognitive organization. We might think of semantic space as "the multifaceted field of conceptual potential within which thought and conceptualization unfold" (Langacker, 1987, p.77); a semantic structure can then be characterized as a location or a configuration within the semantic space. Phonological space is our range of capacity to deal with sounds, with speech sounds as a special case.

When the formation of a grammatical construction is regular to any substantial degree, this regularity is expressed in the grammar by a schematic symbolic unit. The noun *pins*, for example, is a symbolically complex expression formed by integrating the two symbolic units *pin* and *-s*: [[[PIN]/[pɪn]]–[[PL]/[z]]]. This expression instantiates a pattern of plural noun formation embodied in the schematic unit [[[THING]/[...]]–[[PL]/[z]]], where the integration between the two components is exactly analogous to that between the components of *pins* (and of other plural nouns). The schema therefore captures whatever generalization can be made about the nature of the syntactic combination defining the grammatical construction. Figure 2.7 shows the relationship between a schema and an instantiation of a schema.

My model incorporates both semantic units (as meanings of words) and phonological units (as series of phonetic segments comprising words), which include both specific plural forms and schematic units. The model has the capacity to memorize specific forms and to develop schemata that capture the generalizations among the specific forms. Where, then, do these schemata, *rules*, come from? I hoped to have them fall out naturally by giving the model enough exemplars, that is, I wanted to train a network without the benefit of *explicit* rules and wanted to have the schemata represented by the generalizations the network would abstract in the process of learning to produce example words. In a connectionist network schemata are determined by the connection weights between units, which the network develops while trying to produce correct outputs. That is, the rules are built into the associations between forms and meanings. The weights are thus to be learned, not to be presupposed. It is gratuitous to assume that mastery of a rule like N + *-s*, and mastery of forms like *chips* that accord with this rule, are mutually exclusive facets of a speaker's knowledge of his language; it is perfectly plausible that the two might sometimes coexist. We do not lose a generalization by including both the rule and specific plural forms

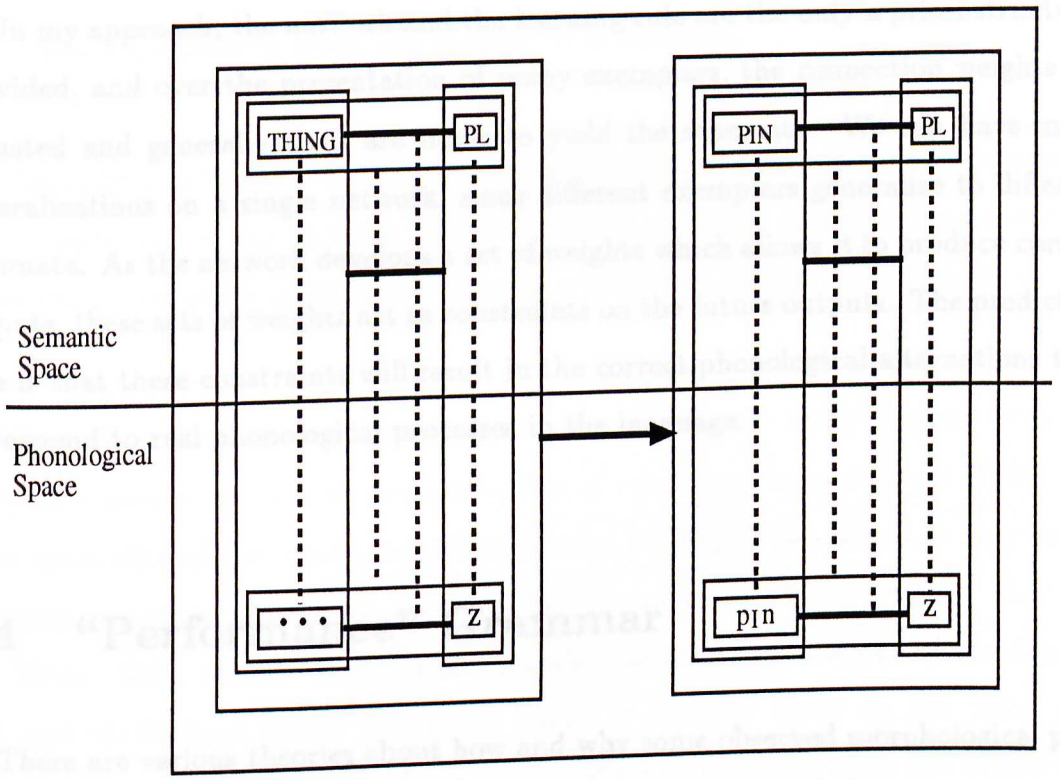

Figure 2.7: The elaborative relationship between a particular composite symbolic structure, *pins*, and the categorizing schema which it instantiates. The internal structure of the schema is exactly parallel to that of its instantiations, specifying how the component morphemes are integrated at the two poles. The schema characterizing the formation of plural nouns with the suffix [z] is therefore identical to the structure for *pins* except that the noun schema replaces the morpheme *pin* as the first component. From Langacker, 1987, p.85.

in the grammar of English, since the rule itself expresses generalization. To claim on an a priori basis that the rule precludes the list, or the converse, is simply to embrace what Langacker calls the *exclusionary fallacy* (Langacker, 1987, p.25).

In my approach, the network and the learning rule are the only a priori structures provided, and over the presentation of many exemplars, the connection weights are adjusted and generalizations are made to yield the schemata. We can have many generalizations on a single network, since different exemplars generalize to different schemata. As the network develops a set of weights which allows it to produce correct outputs, these sets of weights act as constraints on the future outputs. The prediction here is that these constraints will result in the correct phonological alternations that correspond to real phonological processes in the language.

2.4 "Performance" Grammar

There are various theories about how and why some observed morphological phenomena occur in the way they do. However, most traditional theories presuppose abstract underlying representations and a set of *rules* to obtain their surface realizations. Modern generative grammar is based on the notion of "deriving" forms through the application of a series of rules, each of which takes a linguistic representation as input and yields one which is in some sense closer to the "surface". Thus, behind surface forms are underlying representations, abstractions within which each morpheme has an invariant form.

Consider again, as an example, the English plural. The regular plural morpheme takes three different forms, /s/, /z/, and /ɪz/. To account for them in traditional phonology, two rules are invoked:

1. *Voicing Assimilation.* Spread the value of voicing from one phoneme to the next in word final position.

2. *Vowel Insertion.* Word-finally, separate with the vowel /ɪ/ adjacent consonants that are similar in place and manner of articulation.

The first rule is invoked to add /s/ to the nouns which end with voiceless consonants, for example, *hat*, or /z/ to those with voiced final phoneme as with the case of *zone*, while the plural suffix of *page* calls for the second rule due to the similarity of /ʤ/ and /z/.

Most classical symbolists believe that surface forms are really derived from underlying representations with the application of rules. Lachter and Bever (1988) say, "The generality of the application of such rules highlights the fact that they apply to abstract subsets of features, not to actual phonetic or acoustic objects" (Lachter and Bever, 1988, p.202). Pinker and Prince's (1988) critical analysis of the Rumelhart and McClelland (1986) past tense model is crucially based on the claim that the linguistic and developmental facts provide good evidence for rules and underlying representations.

In this thesis, I approach two fundamental problems which are overlooked in both the classical accounts of morphological processes and the RM model (as well as several other approaches that try to overcome the RM model's difficulties), thus answering some of Pinker and Prince's criticisms of connectionist models, as mentioned above.

First, how does knowledge about rules and underlying representations relate to the psycholinguistic processes of production and perception, which relate form and meaning, rather than form to form? The linguistic knowledge implicit in rules and underlying representations is meant to belong to linguistic "competence" and should

thus be shared by both production and perception. Production might to some extent parallel morphological derivations, but perception would be the reverse process. Thus we are confronted with the familiar problem of using rules in one direction when they were designed for another.

A more serious problem, however, occurs in imagining how knowledge about rules and underlying representations might ever be learned. That is, given only surface input forms together with meanings inferrable from context, how is a learner to figure out how the form-meaning relation gets mediated by abstract underlying representations? Where do underlying representations come from? How are rules found and related to each other?

It is customary to assume that a language learner is helped by having certain predispositions about language wired in; however, I begin with an approach which is far more constrained. I assume that the basic building blocks of language acquisition and processing are the simple, neuron-like processing units that connectionist models start out with. What gives such a system its intelligence is its architecture. First, we need some means of representing patterns that take place in time, that is, we need my model to have the capacity to develop a kind of short-term memory that preserves past history.

Second, we need a means of handling both meanings and forms. Throughout this thesis, by "form" I mean a series of phonemes, while by "meaning" I mean the lexical entry of the word in question, together with relevant grammatical features. For example, the word pronounced /hæts/ has the meaning *HAT + PLURAL*. What we need is a mechanism that can incorporate both form (a series of phonemes) and meaning in such a way that the knowledge that is learned is potentially usable in both perception and production tasks.

What I am describing in this section is a rudimentary sort of "performance" grammar; its goals are different from the goals of generative grammar, which is "competence" grammar. I am arguing *not* against rules and underlying representations per se, but against generative/transformational competence rules and underlying representations; I argue *for* the performance rules and underlying representations modeled in a connectionist architecture. For example, the fact that the plural of *top* is pronounced as [tɑps] can be explained in competence grammar as:

```
/tɑpz/  Underlying representation[15]
/tɑps/  Rule 1 as stated above
[tɑps]  Surface Form
```

In contrast, performance grammar does not require any explicit rules and underlying representations. These are to be learned in the process of generalizing over a set of training instances that exhibit the alternations between English singular and plural nouns, for example,

```
TOP + SINGULAR  -->  /tɑp/
TOP + PLURAL    -->  /tɑps/
```

where the items in capitals represent meanings.

I think my approach is more psychologically plausible; and if such an approach can handle perception, production, and acquisition, then generative grammar would become superfluous. I do not believe that each underlying segment goes through a derivation employing morphological rules to produce a "surface" segment. More important is the fact that for speakers meanings trigger the correct phonological/phonetic productions, while for listeners phonetic/phonological material directly evokes the

[15]Here the underlying representation of *plural* is assumed to be /z/.

correct word meanings. An important research question is concerned with the existence of the plausible ways in which people might acquire words by integrating both meaning and form; notably, at what point meaning actually plays its critical role in word acquisition is not clear. It might be the case that both meaning and form should be present from the beginning. Or it might be the case that meaning does not play any critical role at all; its main function is to *aid* in the understanding of words. Between these two opposite ends of the spectrum, there lie infinite possibilities as to when to introduce meaning. In my model meanings are presented from the beginning without any reservations along with the phonetic forms, partly because it was easy to implement, and partly because I believe that both meaning and form should be present from the beginning. This issue will be brought up again in Chapter 6.

Through a series of experiments, I will show that (1) the rules in this system are determined via the connection weights between the units, which the network develops while trying to produce the correct outputs and (2) underlying representations are learned as patterns on the hidden layer, which mediate the relationship between form and meaning.

2.5 Hypotheses

Three hypotheses motivated this project:

Hypothesis 1 A desirable model should accommodate both production and perception in a single network, thus performing the more psychologically plausible task of relating form and meaning, rather than simply associating form to form. Linguistic knowledge is meant to belong to "competence" and should thus be shared by both production and perception.

Hypothesis 2 The model should learn morphological processes without the benefit of underlying representations or *explicit* rules. Given only surface input forms, together with their meanings, the model should discover how the form-meaning relation is mediated. Specifically, rules are to be determined by the connection weights and underlying representations should be reflected in the hidden layer representations. Among the many rule types encountered by the network, it should succeed on rule types which are common in human languages and fail on those which are rare or non-existent.

Hypothesis 3 The model should capture some central generalizations about sound patterns. When the model is trained on a given task involving similar allomorphic variation, it should be able to more easily learn another task involving allomorphic variation. This model should be able to learn rules which are similar to the learned rule more easily than ones which are different from the learned rule. Since hidden representations encode the relationship between form and meaning, a pattern of activity learned by one network should be interpretable by another network.

2.6 Summary

In this chapter, the issues surrounding how words can be perceived and produced have been addressed. Even though a word is a minimal free form in language, it is not the minimal meaningful unit of language, since a given word can often be broken into smaller units called morphemes. All words consist of one or more morphemes, which can be classified in a variety of ways or combined in different ways to create new words. Even though the processes of word formation may vary from one language to another, all languages have the means to create new words and therefore exhibit

the rule-governed creativity that is typical of human languages. Classical symbolic systems postulate explicit rules, as well as innate predispositions in human beings for certain linguistic structures. The study reported here is an attempt to explain morphological processes using a connectionist network and without resorting to any predetermined structure.

Some computational accounts for word perception and production have been attempted, mostly using symbolic processing. Dresher and Kaye's (1990) model for the acquisition of stress systems assumes the existence of preset parameters as a part of the universal grammar where the task of the model is to fix upon the exact value of each parameter. Koskenniemi's two-level morphology is a finite state transducer that processes all rules simultaneously.

It was the Rumelhart and McClelland (1986b) model that opened up connectionist research in word acquisition. The model is a simple two-layer pattern associator that can acquire the markings of the past tense in English. Goldsmith (1991) and Lakoff (1988a) developed a theory which attempts to eliminate the need for rule ordering. Touretzky and Wheeler have developed a connectionist implementation of Lakoff's ideas. Hare's work, Chauvin (1988), and Dorffner (1990) are very similar to my work, reported here. Hare's work addresses, for the first time, the issue of acquisition of phonological regularity in the context of word production. Chauvin (1988) examines empirical results related to first-word acquisition in infants and explores how and why similar phenomena occur in a PDP model. Dorffner (1990) shows how the interpretation and generation of words can be modeled in a connectionist model. Both Chauvin and Dorffner address both comprehension and production of words, very similar to my approach in this thesis.

While most traditional theories presuppose abstract underlying representations and a set of *rules* to obtain surface realizations, my approach does not presuppose

any underlying representations or rules. The rules are to be learned by the network. By accommodating both production and perception in a single network (Hypothesis 1), the model should learn morphological processes without the benefit of underlying representations or *explicit* rules (Hypothesis 2) and can capture some of the central generalizations about sound patterns (Hypothesis 3).

3

Connectionist Models

In the last chapter, the problem of perceiving and producing words was addressed. Since the model I used in this project was based on a connectionist model, I will take a closer look at connectionist learning procedures in this chapter. Also different approaches to temporal processing will be reviewed.

3.1 Connectionist Learning Procedures

In Chapter 1, a brief introduction to connectionism was given. Connectionism is an approach to cognitive modeling which assumes that knowledge is represented by a parallel, distributed information processing structure consisting of processing units densely interconnected via unidirectional weights. Each unit has a single output which fans out into many other units through its connections. Typically there are both input units, which receive input signals from the outside world, and output units, which represent the system's response to that input. There may be one or more "hidden" unit layers between the input units and the output units. Processing

involves activating the input units; this activation is spread through the connections to produce a pattern of activation on the output level. This pattern is compared to a "desired" output and the discrepancy is calculated to adjust the weights.

Consider a network that has input units which are directly connected to output units. By minimizing the error measure (mean squares) between actual outputs and "desired" outputs, we are guaranteed to find the set of weights that gives the desired output. This learning procedure is the "least mean squares" (LMS) procedure (Widrow and Hoff, 1960; Rosenblatt, 1962).

The "back-propagation" learning rule (Rumelhart, Hinton, and Williams, 1986), one of the most commonly used weight updating rules, is a generalization of the LMS rule that works for networks with hidden units. A variant of this procedure was discovered independently by Werbos (1974), Le Cun (1985) , and Parker (1985). In this learning procedure discrepancy is back propagated to adjust the weights as following:

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pj}$$

where $\Delta_p w_{ij}$ is the amount of change to be made on the connection weight w_{ij} for pattern p, η is a proportionality constant, called the *learning rate*, o_{pj} denotes the output of the jth unit for the pattern p, and δ_{pj} represents the error signal for the unit j given pattern p, which is computed recursively starting from the output units. If a unit is an output unit, its error signal is computed as

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj})$$

where $f'_j(net_{pj})$ is the derivative of the activation function which maps the total input to the unit to an output value. The error signal for the hidden units, for which no specified targets exist, is determined in terms of the error signals of the units

to which it directly connects as well as the weights of those connections with the following formula:

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}.$$

In order to use this learning rule, we need to know the derivative of the activation function of unit u_j, $f'_j(net_j)$. For the most commonly used *logistic* output function given in Chapter 1, the derivative is computed as

$$\frac{\partial o_{pj}}{\partial net_{pj}} = o_{pj}(1 - o_{pj}).$$

Thus, the error signal for an output unit is given by

$$\delta_{pj} = (t_{pj} - o_{pj}) o_{pj}(1 - o_{pj}),$$

and the error for an arbitrary hidden unit is given by

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{kj}$$

where t_{pj} is the target input for jth component of the output pattern for pattern p. This rule requires only that changes in weight be proportional to $\partial E_p / \partial w$ where E_p is the error measure on input/output pattern p. The constant of proportionality is the learning rate in the equation. The larger the constant, the larger the changes in the weights. For all practical purposes, we want this learning rate to be as large as is possible without it leading to oscillation. One way of achieving this is to introduce a new term, called *momentum*, in the equation, yielding a new rule:

$$\Delta w_{ij}(n+1) = \eta(\delta_{pj} o_{pj}) + \alpha \Delta w_{ij}(n)$$

where the subscripts n indexes the presentation number and α is a constant which determines how much past weight changes will contribute to the current weight change.

3.2 Learning Temporal Processes

When designing the model, I considered its capability to accommodate temporal processing as one of the most important criteria, since the morphological processes that I am studying in this thesis are temporal ones. Would a simple feed-forward network be sufficient for the tasks in this research? Or would it be absolutely necessary to use a recurrent network? If a recurrent network was not indispensable, a simple feed-forward network would be preferable, since recurrent networks are much more complicated and in general take more time and space to run. However, a recurrent network was necessary, since without some form of short-term memory (STM) to store the previous events, the network could not distinguish two identical phonemes that might appear in the same word, for example, the two /b/'s in *Bob*, and it would be impossible for the network to decide when to apply a morphological rule to the word in question. The path the network takes to encode the internal representations is very important to the current study. The system cannot know how to behave only on the basis of the current input; the previous context is essential in perception and production of words. In other words, the network has to have some way of knowing "where" it is in the temporal sequence (see Port, 1990, on the issue of representation of temporal patterns). In this section, I will briefly review some of the techniques in connectionist framework that address the learning of time sequences. Hertz, Krogh, and Palmer (1991) classify the learning of sequences into three distinct tasks: *sequence recognition*, where a particular output is sought given a specific input sequence, *sequence reproduction*, where the network must be able to generate the rest

of the sequence when it sees only parts of it, and *temporal association*, in which a particular output sequence must be produced in response to a specific input sequence. My task is very much in the domain of temporal association. Each of the approaches discussed below can learn one or more of these tasks.

3.2.1 Simple Feed-forward Networks

The simplest way to perform sequence recognition is to turn the temporal sequence into a spatial pattern. One way of achieving this is by explicitly coding discrete states into the unit functions which can change over time such as did Feldman and Ballard (1982). In their network,

> each unit will be characterized by a small number of *discrete states* [italics added] plus:
> p—a continuous value in [-10,10], called *potential* (accuracy of several digits)
> v—an *output value*, integers $0 \leq v \leq 9$
> i—a vector of *inputs* $i_1, \ldots, i_n$ (p.211).

Another way of achieving sequence recognition using a simple feed-forward network is to encode temporal information as part of the input. For example, Rumelhart and McClelland's (1986b) past tense model adopts Wickelfeatures (see Wickelgren, 1969, for context-sensitive coding) to allow for temporal context sensitivity. According to this scheme, a string is represented as a set of three-character-sequences which it locates. RM call such trigrams *Wickelphones*. For example, a word like *net* translates to $\{{}_{\#}n_e,\ {}_{n}e_t,\ {}_{e}t_{\#}\}$. In this approach, a 'word-boundary (#)' is a character, too. A serious problem with this representational scheme is that distinct lexical items

may be associated with identical representations[1] as illustrated in Pinker and Prince (1988):

> For example, the Australian language Oykangand (Sommer, 1980) distinguishes between *algal* 'straight' and *algalgal* 'ramrod straight', different strings which share the Wickelphone set alg, al#, gal, lga, #al, as can be seen from the analysis in (5):

(5) a. algal

#al

alg

lga

gal

al#

b. algalgal

#al

alg

lga

gal

alg

lga

gal

al#

(p.97)

The multilayer feed-forward networks of type as shown above cannot perform such temporally extended tasks as the recognition of the above two distinct sequences, since they necessarily go to a stationary state and thus produce static mappings.

[1]Mozer (1990) reports on a connectionist network that discovers *faithful* 'Wickelfeature' representations – ones that do not lose critical information about the sequence to be encoded, thus overcoming this problem.

3.2.2 Moving Window Systems

Most connectionist models of sequence recognition use the "moving window" paradigm (Elman and Zipser, 1988; Lang et al., 1990; Sejnowski and Rosenberg, 1987; Tank and Hopfield, 1987; Waibel et al., 1988). Typically, these models use one pool of input units for the event at time t, another pool for the event at $t+1$, and so on. The basic idea is to have a separate set of nodes to represent the set of raw inputs from each time slice. For example, Elman and Zipser's (1988) model was presented with 64 ms speech samples in 20 slices. For each slice, 16 normalized spectral energy measures were provided, yielding a network with 320 input nodes. Elman and Zipser showed that this sort of network can be trained to successfully categorize stop-vowel syllables despite the variation inherent in the hundreds of tokens that were taken from a single speaker. At first glance, networks of this type appear to capture useful information about location in time, in terms of spatial organization within the window. But from the standpoint of the network, the nodes representing each time slice are just parallel input channels. They do not even have an intrinsic serial order.

A variant of the time window approach described above has inputs that arrive at different places in the network through time-delay connections: only one event is presented at a time, but each input unit has a set of connections coming out of it all with different time delays as shown in Figure 3.1, making it possible for a higher layer of units to have access to inputs that appeared on an earlier time step. This type of architecture is sometimes called **time-delay neural networks**. In Waibel et al.'s (1988) network, there are 16 input units which serve as spectral coefficients. Each unit in the first hidden layer receives input (via 48 weighted connections, that is, 16 coefficients over three frames with time delay 0, 1, and 2) from the coefficients in the 3 frame window. They report that their network performed very well on the task of speaker-dependent recognition of the voiced stops /b/, /d/, and /g/, extracted from

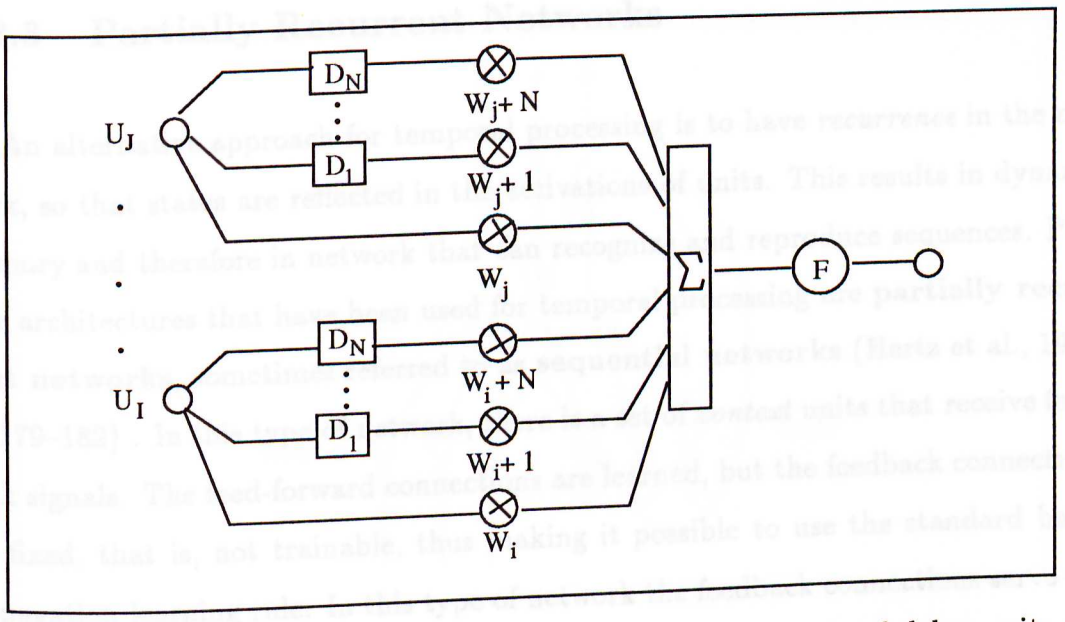

Figure 3.1: A Time Delay Neural Network unit. The J inputs of delay units are multiplied by several weights, one for each delay and one for the undelayed input.

various phonetic contexts. Because the time delays are fixed in the network and the STM consists of unanalyzed input events, this approach is quite similar to the window approach.

There are several drawbacks to the moving window approaches. First, a moving window approach is not psychologically plausible, since "it requires that there be some interface with the world, which buffers the input, so that it can be presented all at once" (Elman, 1990, p.181). Another problem with such an approach is that it imposes a rigid limit on the duration of the input window. The size of the window must be chosen in advance to accommodate the longest possible sequence. As a result, computing resources are wasted, since some of the unused pools of units must be kept available for the rare occasions when the longest sequences are presented. Finally, the input signal must be aligned properly and must arrive at exactly the correct rate. Therefore, the recognition task and the generation of sequential patterns are not easily handled using simple tapped delay lines.

3.2.3 Partially Recurrent Networks

An alternative approach for temporal processing is to have *recurrence* in the network, so that states are reflected in the activations of units. This results in dynamic memory and therefore in network that can recognize and reproduce sequences. Popular architectures that have been used for temporal processing are **partially recurrent networks**, sometimes referred to as **sequential networks** (Hertz et al., 1991, pp.179–182) . In this type of network, there is a set of *context* units that receive feedback signals. The feed-forward connections are learned, but the feedback connections are fixed, that is, not trainable, thus making it possible to use the standard back-propagation learning rule. In this type of network the feedback connections serve the purpose of STM. STM is held by the context units, which are copies of a particular layer from the previous time step. Thus at any given time step, the network receives not only the current input but also some memory of its previous state. But because the previous input was also a function of the previous STM pattern, the network encodes information over many previous time steps. Another advantage of recurrent networks over simple feed-forward networks is that they avoid the difficulty of a fixed-length representation of variable-length patterns, since the length of input patterns is not explicitly represented in the network architecture. In a window representation, the size of the input layer dictates the size of the window; but in a dynamic network events are processed one at a time, thus eliminating the difficulty of representing a variable-length input within a fixed-length input layer.

Decay-based Recurrent Networks

In Jordan's (1986a, 1986b) approach (see Figure 3.2), the output vector of the network is linearly averaged into a context vector, which is given to the network as a

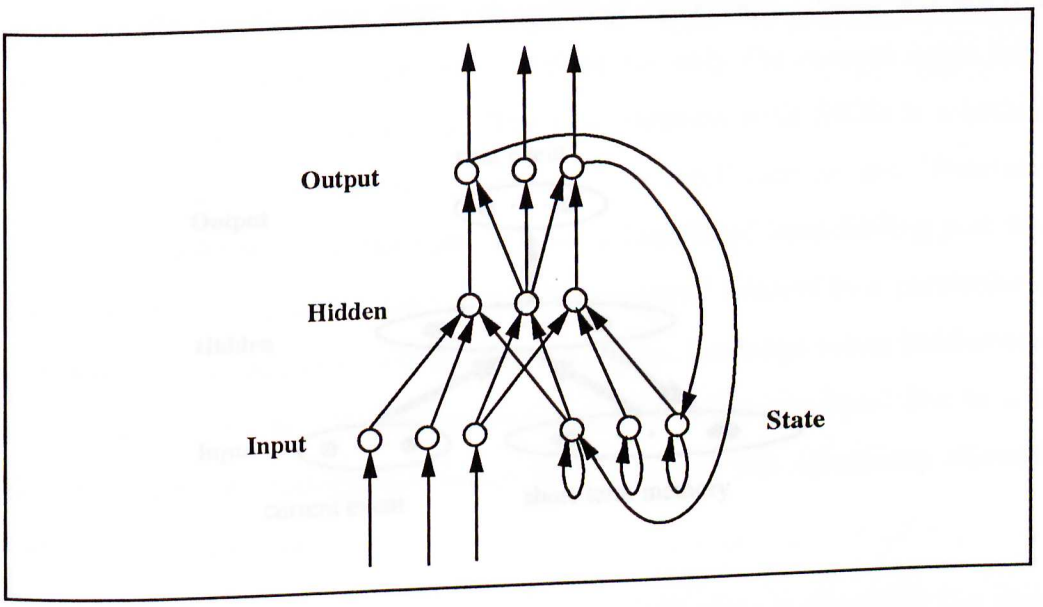

Figure 3.2: A Decay-based Recurrent Network.

part of the input. We will refer to this type of networks as **decay-based recurrent networks**. In this type of network, input consists of two parts. The first part, labeled the *plan*, is an arbitrary representation of the sequence to be produced and remains constant throughout the processing. The second part of the input is a bank of context units called the *state*. The context vector at time t is some proportion (μ) of its value at time $t-1$ (arrows pointing to the same unit in the figure), plus the output vector at time $t-1$. This serves as a temporal context and aids the system in determining what part of the sequence is the next to be produced. As a result, the network has an exponentially decaying representation of its history. This kind of network can be trained to produce nearly arbitrary sequences.

Simple Recurrent Networks

Elman (1990) developed an architecture where, in addition to the usual input, hidden, and output units, there is a set of context units that hold a copy of the

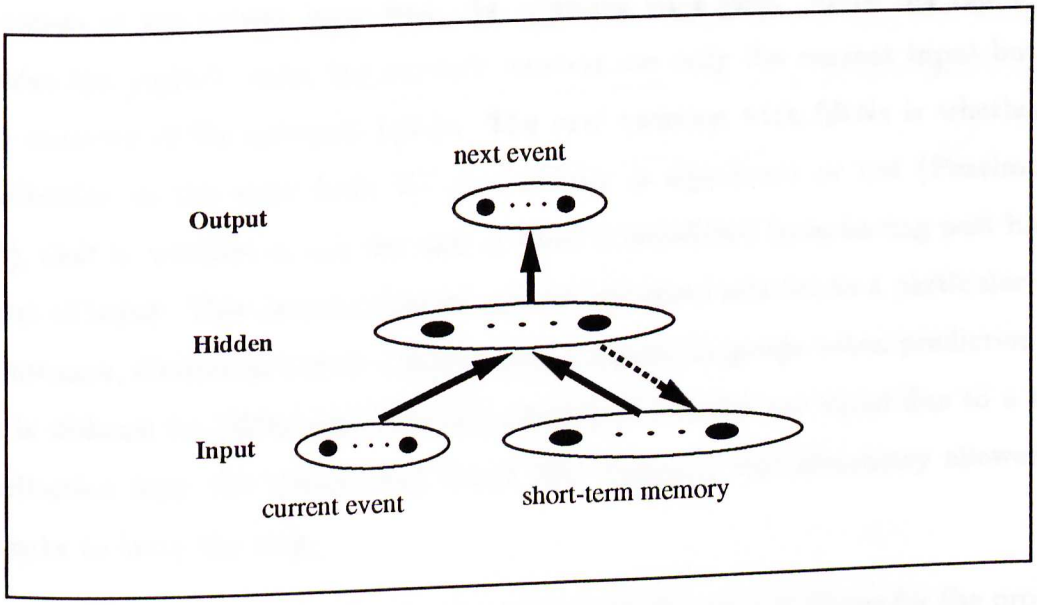

Figure 3.3: A Simple Recurrent Network.

hidden layer activations from the prior time step. These networks have been called **simple recurrent networks** (SRNs). The basic structure of an SRN is shown in Figure 3.3. SRNs differ from decay-based recurrent networks by having input to the context units fed from the hidden layer (instead of the output layer) and by lacking of self-recurrency. The hidden units have the task of mapping the input to the output, and because the input now includes their own prior states, they must develop representations which serve as some form of STM. Note that the context units are part of the input layer and thus there is no need to calculate $\partial C_k / \partial w_{ij}$ to adjust the weights, where C_k is the k-th context unit, making it possible to apply standard back-propagation without any modifications in training the network. SRNs are usually trained on prediction tasks: given an input event, the network is to indicate the next event.[2] SRNs maintain past history in their context units, which

[2]Of course, this does not mean SRNs cannot be used for other tasks, including autoassociation and sequence recognition (Stolcke, 1990), simply that SRNs have mostly been used for prediction tasks.

are copies of the hidden layer from the previous time step. Since the input layer includes the context units, the network receives not only the current input but also some memory of the previous inputs. The real question with SRNs is whether the contribution to the error from the past history is significant or not (Pearlmutter, 1990), that is, whether or not the task at hand is benefitted from having past history as part of input. This question can be only be answered relative to a particular task. For instance, Cleeremans et al. (1989) found a regular language token prediction task that is difficult for SRNs when the transition probabilities are equal due to a small contribution from the history, but found that breaking this symmetry allowed the networks to learn the task.

An SRN is particularly suitable for language study, since it allows for the processing of serial inputs. Thus, language can be processed naturally, element by element. Furthermore, an SRN makes minimal assumptions about special knowledge required for training. When used for prediction tasks such as those examined in this thesis, SRNs provide a teacher which is readily available in the environment on the next time slice, that is, the teacher and the input are of the same form. Thus there are no a priori theoretical commitment as to what type of output the model is expected to produce, which might otherwise bias the outcome. This is different from the kind of architecture which maps sequential language input onto non-sequential feature-based semantics (Stolcke, 1990).

Elman and others (Cleeremans et al., 1989; Cottrell and Tsung, 1989; Harris and Elman, 1989; Servan-Schreiber et al., 1988; Servan-Schreiber et al., 1989; Small, 1990; Stolcke, 1990) have used SRNs for various tasks, mostly training them on prediction. Exclusive-OR, structure in letter sequences, representation of vocabulary, discovering lexical classes from word order, and pronoun resolution are the experiments reported in Elman (1990). Elman (1989) addresses the nature of representations in

connectionist models by simulating problems in the distinction of *type* and *token*, the representation of *lexical categories*, and the representation of *grammatical structure*.

Servan-Schreiber, Cleeremans, and McClelland (1988, 1989) used an SRN to predict successive elements of sequences in finite state automata. They found that when the network was trained with strings from a particular finite-state grammar, it could learn to be a perfect finite state automaton for the grammar in question. They report that the network encodes prediction-relevant information about the path on the hidden layer and illustrate the phases of learning by the network.

Harris and Elman (1989) address the issues of how an SRN might represent co-occurrence relationships, such as those holding between a script setting (e.g., "clothing store") and a script item ("shirt"), or the relationships that specify the feature match between the gender of a pronoun and its antecedent.

Cottrell and Tsung (1989) trained an SRN and a decay-based recurrent network and compared them on the task of adding two multi-digit numbers. They found that networks of this type can learn simple programming constructs that are not nested. They also report that a decay-based recurrent network cannot "remember" things about its input that are not reflected in its output, since it has access to only current input and the output history and keeps no record of previous inputs or the internal states of the system.

Small (1990) studied lexical ambiguity resolution and semantic parsing. He reports that the network learned to interpret correctly the intended meaning of the words "take out" in context. He also reports that the network predicted a pronoun of the correct gender in the appropriate contexts. Furthermore, analysis of the hidden layer revealed that the network acquired a representation of the contextual information; the syntactic and semantic structures of language, such as the linguistic notion of

"noun", or the interesting and useful notion of "the word *out* in the context of a female agent", are represented in a distributed non-symbolic form. This work is relevant to my study, since Small showed that SRNs can learn to perform linguistic tasks without any explicitly coded pre-existing linguistic knowledge and also demonstrated that the network can create a distributed internal representation of various heuristically useful concepts.

Stolcke (1990) investigated the possibilities of using SRNs as transducers to map sequential natural language input onto non-sequential feature-based semantics. The networks performed well on sentences containing a single main predicate and multiple-feature objects. The network was also able to process multiple levels of sentence-final embedding but only one level of center-embedding due to its inability to retain information that is not reflected in the outputs over intermediate phases of processing.

In preceding paragraphs, I sketched how other researchers used SRNs to investigate their power and usefulness in various language studies without any explicit representation of linguistic knowledge in the form of rules. My model differs from other SRNs in that it is trained on autoassociation as well as prediction. That is, there are two sets of output units, one for the current event and another for the next event. This is one way to force the network to distinguish the different input patterns on the hidden layer as suggested by Servan-Schreiber et al. (1989).

3.2.4 Back-propagation Through Time

We have just discussed networks that were partially recurrent. Now let us turn to **fully recurrent networks**, where any unit can connect to any other unit including

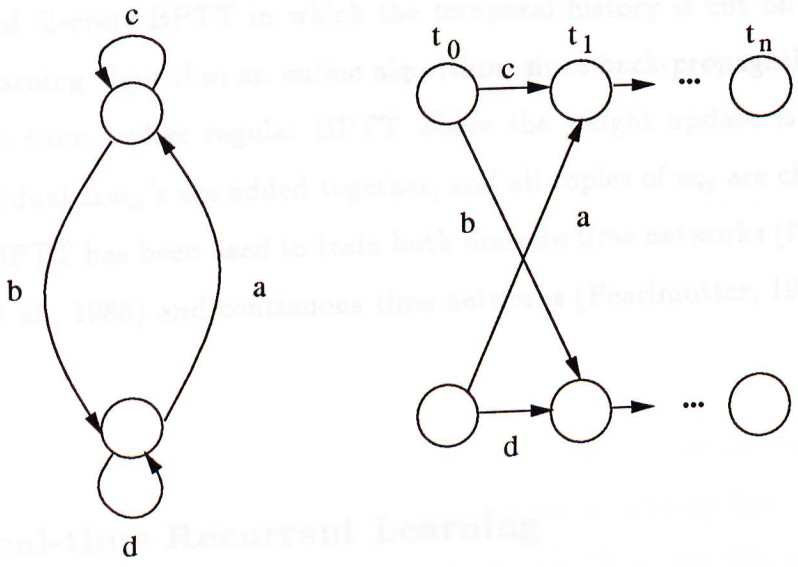

Figure 3.4: A simple iterative network in which stimulation for different time frames is supplied to different input nodes. The single kernel network on the left is repeated for each time slice. The flow of activations through the network simulates recurrent activity through time. Reproduced from Port, 1990, with the permission of the author.

itself. Fully recurrent networks are most general and can perform temporal association tasks including sequence recognition and reproduction. If we are interested in sequences of a finite length T, we can use back-propagation through time (BPTT). As described in Rumelhart et al. (1986), the back-propagation learning algorithm can be used to train a recurrent network, since any recurrent network can be "unrolled" in time to yield a feed-forward network that has a layer for each time step, as illustrated in Figure 3.4.

BPTT is a technique to learn non-fixpoint attractors and to produce the desired temporal behavior over a bounded interval in fully recurrent networks. This can be done with minor modifications to the standard back-propagation algorithm. While standard back-propagation would normally yield different increments Δw_{ij} for each time step, BPTT must retain identical w_{ij}'s. Note that the SRN's learning algorithm

is a version of discrete BPTT in which the temporal history is cut off. This makes the SRN's learning algorithm an online algorithm, since back-propagation is done at each point in time, unlike regular BPTT where the weight update is done offline; that is, individual Δw_{ij}'s are added together, and all copies of w_{ij} are changed at the same time. BPTT has been used to train both discrete time networks (Nowlan, 1988; Rumelhart et al., 1986) and continuous time networks (Pearlmutter, 1989).

3.2.5 Real-time Recurrent Learning

There are other learning algorithms for fully recurrent networks (see Pearlmutter, 1990, and Williams and Zipser, 1990, for excellent overviews). For example, an algorithm proposed by Williams and Zipser (1988, 1989a, 1989b) performs the task of encoding STM better than BPTT. While BPTT uses the backward-propagation of error information to compute the error gradient, their algorithm, called *real-time recurrent learning* (RTRL) propagates activity gradient information forward. The algorithm is stated as follows:

> [The algorithm] create[s] a dynamical system with variables $\{p_{ij}^k\}$ for all $k \in U, i \in U$, and $j \in U \cup I$, and dynamics given by
>
> $$p_{ij}^k(t+1) = f_k'(s_k(t)) \left[\sum_{l \in U} w_{kl} p_{ij}^l(t) + \delta_{ik} z_j(t) \right],$$
>
> with initial conditions
>
> $$p_{ij}^k(t_0) = 0,$$

and it follows that

$$p_{ij}^k(t) = \frac{\partial y_k(t)}{\partial w_{ij}}$$

for every time step t and all appropriate i, j, and k (Williams and Zipser, 1989, p.273).

In this algorithm, U denotes the set of indices for outputs in the network, I denotes the set of indices for external inputs, f_k indicates the squashing function, s_k is the net input to the kth unit, w_{kl} is the weight from the lth to the kth unit, z_j is the concatenation of input and output, and y_k is the output of the units. Thus we may use the above two equations to compute the quantities $p_{ij}^k(t)$ at each time step in terms of their prior values and other information, depending on the activity in the network at that time. Combining these values with the error vector $\mathbf{e}(t)$ for that time step via the equation

$$\frac{\partial J(t)}{\partial w_{ij}} = \sum_{k \in U} e_k(t) p_{ij}^k(t)$$

yields the negative error gradient $\nabla_{\mathbf{W}} J(t)$. Because the p_{ij}^k values are available at time t, the computation of this gradient can occur in real time.

Williams and Zipser propose a hybrid strategy involving both RTRL and back-propagation to be used for an SRN (Williams and Zipser, 1990). In this approach, the p_{ij}^k values need only be stored and updated for the hidden units, with back-propagation used to determine other necessary quantities. The error gradient is computed by means of the following equation,

$$\frac{\partial J(t)}{\partial w_{ij}} = \begin{cases} \delta_i(t) x_j(t) & \text{if } i \in U_O \\ \sum_{l \in U_H} \varepsilon_l(t) p_{ij}^l(t) & \text{if } i \in U_H, \end{cases}$$

where U_O denotes the set of indices of units in the output layer and U_H is the set of indices of units in the hidden layer; $\delta_i(t)$ is obtained by back-propagation entirely within the hidden layer. The p_{ij}^k values, for $k \in U_H$, are updated by means of the equation

$$p_{ij}^k(t) = f_k'(s_k(t)) \left[\sum_{l \in U_H} w_{kl} p_{ij}^l(t-1) + \delta_{ik} z_j(t-1) \right],$$

which is just the RTRL equation specialized to take into account the fact that w_{kl} is 0 if $l \in U_O$.

The problem of RTRL is that it is computationally expensive and that it requires non-local information. In general, the "minimal number of p_{ij}^k values needed to store and update for a general network having n units and r adjustable weights is nr. For a fully interconnected network of n units and m external input lines in which each connection has one adaptable weight, there are $m^3 + mn^2$ such p_{ij}^k values" (Williams and Zipser, 1989, p.275). For the hybrid system described above, the average time required per time step is $\Theta(nw_U + nw_A + n^2 w_A)$, where w_U denotes number of nonzero weights between units, w_A number of adjustable weights, and n denotes the number of units. Worst-case complexity shows that it requires $\Theta(n^3 + n^4)$. On the other hand, BPTT used in an SRN requires an average time per time step of only $\Theta((w_U + w_A)/2)$ and in the worst case, the order of magnitude of time required for the BPTT is $\Theta(n^2)$ (see Figures 1 and 2 in Williams and Zipser, 1990, p.37).

Overall, the RTRL algorithm is probably the best in terms of performance, since it can encode STM better than any other learning algorithms. But for many temporal processing tasks, including the tasks explored in this thesis, partially recurrent architectures with context units suffice, and are much less costly computationally. The ability to represent the internal state of the system is critical to my experiments. The

model should be able to encode the path taken to achieve the given output. Without this information, the model cannot encode a morphological rule: it must know, for example, which suffixes can follow which phonemes. Simply preserving the past history of the outputs as can be done in any decay-based recurrent network may not be enough to sufficiently encode path information critical to encoding morphological rules.[3] Since decay-based recurrent networks cannot represent the internal states of the system (Cottrell and Tsung, 1989), I decided to use an SRN in this thesis.

3.3 Summary

In the study reported in this thesis, a recurrent connectionist network was used to conduct experiments on the perception and production of words. A connectionist network is a parallel, distributed information processing structure consisting of processing units which are interconnected via unidirectional weights. Each unit typically has many input signals which calculate a single output signal, which then fans out to many other units via connections. There are three different types of units: input, hidden, and output. Processing involves activating input units; this activation spreads through the connections to produce a pattern of activation on the output level. This pattern is compared to a "desired" output and the discrepancy between the two is calculated and the weights adjusted accordingly. The back-propagation learning rule is a typical representative of a learning procedure that minimizes mean squares between actual outputs and "desired" outputs.

[3]This is still a working hypothesis, since I have not tested a decay-based recurrent network to determine its inadequacy for my experiments. As will be shown in Chapter 5, Gasser (personal communication) showed that SRNs perform better than other types of networks in learning the "rote" aspects of the processes.

When designing the model, I considered the capability to accommodate temporal processing to be one of its most important features. Since the morphological processes that I am studying in this thesis are temporal, I need to have some form of short-term memory in which to store the previous events. There are many techniques in the connectionist framework that address the learning of time sequences. Most connectionist models of sequence recognition use the "moving window" paradigm. These models use a pool of input units for the event at time t, another pool for the event at $t+1$, and so on. A variant of the time window approach employs inputs that arrive at different places in the network via time-delay connections. An alternative approach for temporal processing is to build *recurrence* into the network, so that previous states are reflected in the activations of certain units. Jordan (1986a) and Elman (1990) have developed partially recurrent networks, where a set of *context* units receives feedback signals. The feed-forward connections are learned, but the feedback connections are fixed, thus making it possible to use the standard back-propagation learning rule. For more general tasks, there is the fully recurrent network, in which any unit can connect to any other unit including itself. Back-propagation through time and real-time recurrent learning are examples of training algorithms for this type of network.

4

The Model

In this chapter, a detailed description of the system used for the project will be reported. I will also argue that the model chosen here is appropriate for and capable of the tasks reported in this research.

4.1 System Overview

I used a relatively constrained three-layer network, one in which feed-forward connections are supplemented by limited feedback connections. Figure 4.1 shows the network architecture used for all of the experiments.

The input layer consists of three cliques[1] of units: the Form clique, the Meaning clique, and the Context clique. Depending on the task, the Form clique in the input and output layers consists of 8 or 13 units representing a phonological segment. Each Meaning clique consists of 7 units, 6 of which represent a stem meaning (hereafter

[1] A clique is a collection of units with a common function.

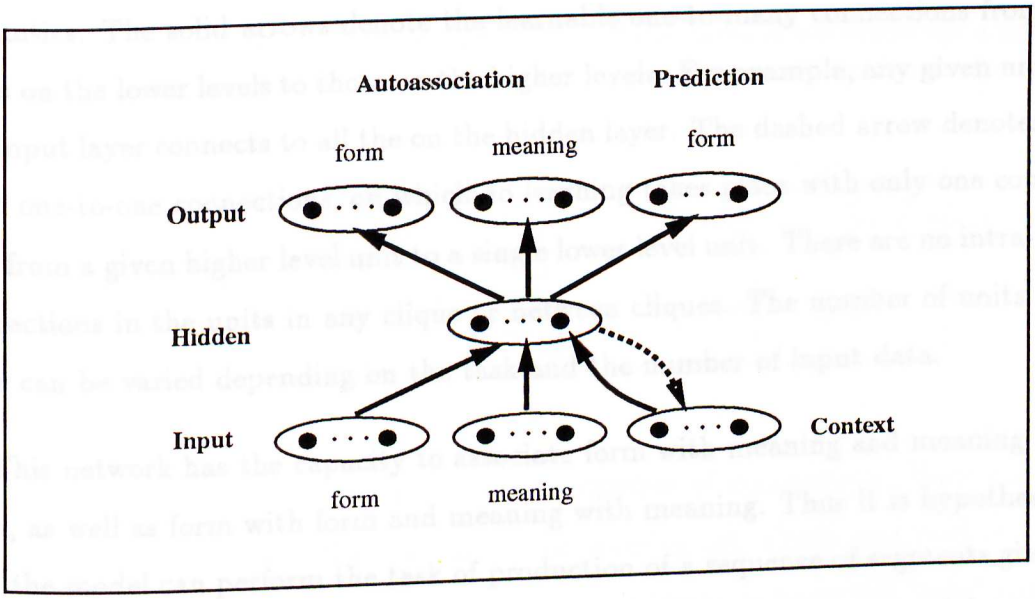

Figure 4.1: Architecture of the network used in the morphological rules study.

referred to as *s-meaning*) and 1 of which represents the grammatical "number"[2] (hereafter *g-number*; by *meaning*, I refer to s-meaning and g-number together) of the word. The network has a variable number of hidden units and an equal number of Context units. Each of the first two cliques receives input from the outside, while the Context units receive a copy of the activations on the hidden layer from the previous time step. The hidden units receive activations from the input layer, feeding the output layer. In addition, the activations on the hidden layer are fed back to the Context units. The output layer receives activations from the hidden layer. The output layer produces outputs in accordance with the current form and meaning and predicts the next input form. Given the current form and meaning, the network is trained to replicate them on one part of the output layer (autoassociation) and to predict what comes next in the sequence (prediction). My concern is with the arbitrary relationship between form and meaning; hence we need not concern ourselves in this thesis with genuine

[2]This grammatical feature unit can be used for any grammatical feature, including tense and number. It is for convenience that I call it *number*.

semantics. The solid arrows denote the learnable one-to-many connections from the units on the lower levels to those on the higher levels. For example, any given unit on the input layer connects to all the on the hidden layer. The dashed arrow denotes the fixed one-to-one connections, on which no learning takes place with only one connection from a given higher level unit to a single lower level unit. There are no intra-level connections in the units in any clique or between cliques. The number of units on a layer can be varied depending on the task and the number of input data.

This network has the capacity to associate form with meaning and meaning with form, as well as form with form and meaning with meaning. Thus it is hypothesized that the model can perform the task of production of a sequence of segments given a meaning, or of a meaning given a sequence of segments. It has the potential to make generalizations across morphologically related words.

4.2 Why This Architecture?

The model employed in this project is a slightly modified version of the SRNs discussed in the previous chapter. There are many different network architectures and learning algorithms available for use, as shown in Chapter 3. Why then did I choose this particular architecture? There are theoretical as well as practical reasons.

4.2.1 Experiments on Different Architectures of the Network

Gasser (personal communication) ran experiments on the best network types for perception and production tasks. In each case, he tried out various architectures based on the network used in this thesis, using the same problem with the same

Table 4.1: Results of experiments on different architectures of networks for a perception task. Numbers indicates the percentage of correct output. From Gasser (personal communication).

	Autoassociation		No Autoassociation	
	Prediction	No Prediction	Prediction	No Prediction
No meaning copy	.24	.31	.17	.20
Copy, $\mu = 0$	.22	.32	.13	.15
Copy, $\mu = .5$	.26	.32	.12	.12

hidden layer size but obviously not the same number of connections. Each network was run for 40 epochs and then tested on the training set, which consisted of words of only one morpheme, so only the ability of the networks to learn the "rote" aspects of the processes was tested. In each run, there were 30 words, in which number of phonetic features (and hence possible segments) varied from 5 to 7. 3-segment, 4-segment, and 5-segment words (+ word boundaries) were distributed equally in groups of 10, which were otherwise randomly generated.

Results for the perception task are summarized in Table 4.1. For the perception task, all the networks had context layers with copies of the hidden layer activations from the prior time step. There were three variants. The first variant, dealing with meaning, had three cases: no copy of meaning output, meaning output copied with $\mu = 0$ (no memory except for the last output)[3], and copy with $\mu = .5$. In the second variant, the network performance was compared with and without the task of autoassociation (output current segment). In the third variant, the network performance was compared with and without prediction (output next segment).

The results for the production task are shown in Table 4.2. For this task, all

[3]This proportionality constant is the decay factor that tells how much the previous output value should contribute to the current activation value.

Table 4.2: Results of experiments on different architectures of networks for a production task. From Gasser (personal communication).

	Autoassociation		No Autoassociation	
	Context	No Context	Context	No Context
$\mu = 0$	.75	.46	.77	.43
$\mu = .5$	.78	.51	.80	.58

the networks copied output via connections from the prediction output to the input segment layer. There were three variables: (1) $\mu = 0$ and $\mu = .5$, (2) with and without context layer for the copy of previous hidden activations, and (3) with and without autoassociation.

Two things seem clear: (1) For production, having a context layer helps, and (2) for perception, autoassociation helps. By looking at these results, I can safely say that the architecture chosen for the studies conducted in this thesis is justified. In what follows, I will describe why integrating all of these architectures (SRN, autoassociation, and prediction) helps the network learn to perceive and produce words.

4.2.2 Advantages of Simple Recurrent Networks

It has already been shown in Chapter 3 why SRNs were used as the basis for my model. SRNs have the ability to learn the kind of temporal processing that is a prerequisite to the perception and production of words by storing the past network history on the context units. Computationally, this is much less costly than the networks employing RTRL algorithm. The version of STM overcomes the disadvantage of using a decay-based recurrent network. In a decay-based recurrent network, the state to be retained by the network across time must be present in the output, since

the network cannot remember about its input that is not reflected in its output. The SRN, on the other hand, stores a copy of the transformed version of the input which is available at the hidden layer on the context layer. This copy is recycled on each time step, whereby the SRN should be able to remember input that is not necessarily reflected in its output.

4.2.3 The Role of Autoassociation

As noted in the previous section, my model differs from other SRNs in that it is trained on autoassociation as well as prediction. This is a way to force the network distinguish different input patterns on the hidden layer, a prerequisite to the system's making use of the hidden layer as an STM. Servan-Schreiber et al. (1989) report that

> Encoding of sequential structure in the patterns of activation over hidden layers proceeds in stages. *The network first develops stable hidden-layer representations for the individual letters* [italics added], and then for individual arcs in the grammar. Finally, the network is able to exploit slight differences in the patterns of activation which denote a specific path through the grammar. (p.651)

Developing stable hidden layer representations for individual input patterns is necessary for my model, since my experiments call for it to have the ability to represent all inputs up to a given point in time; the context is needed in order for the network to be able to predict the next segment.

4.2.4 The Role of Prediction

Elman (1989) has shown that, given words as input and trained on the simple, unsupervised task of predicting the next word, SRNs can learn a great deal about sentence structure. My current research focuses on what can be expected from such a network, given a single phonological segment at a time and trained to predict the next phonological segment. If a system could learn to do this successfully, it would have a version of what phonologists call *phonotactics*,[4] at least a left-to-right sort of phonotactics; that is, it would have knowledge of what phonological segments tend to follow other phonological segments in a given context.

But what does this sort of knowledge have to do with the actual perception and production of words? This is related to the general question of what elements the processes of perception and production share. One could argue that what the two processes share is precisely the left-to-right phonotactics that they can get by training an SRN on the prediction task. In perception, knowledge about phonotactics in general *might* provide a basis on which the system could learn the representations of the particular words that it is to perceive. For example, suppose that every /b/ is followed by a vowel in a particular language. Once a /b/ is perceived, the system can then predict a vowel with confidence, thereby narrowing the set of possible syllables, and, eventually words, to perceive. In the case of production, the system is trying to decide which phonological forms to produce following the current one, that is, the one it has just produced. This is in fact exactly how the system was trained in both

[4]Phonotactics refers to the constraints a language has in terms of which sound is allowed in the context of other sounds, that is, the ordering (tactics) of phonemes. For example, in English there are no words that begin with *stl-* or *spw-*. This is due to a constraint in English preventing two initial consonants with the same place of articulation, if the first consonant of the two is a stop (O'Grady et al., 1989, p.56).

perception and production tasks. Since both word perception and production supposedly obey the speaker/listener's phonotactic knowledge of the language (Church, 1987), training on the phoneme prediction task might provide a way of integrating the two processes within a single network, allowing the network to take advantage of known phonotactic constraints in order to restrict the set of possible syllables – and, thus, possible words.[5]

4.2.5 Form Units and Meaning Units

The model described in this chapter incorporates both Form units and Meaning units. Since my model should perform both perception and production tasks, it should have both kinds of information available in the system. It has to have the ability to associate form with meaning as well as meaning with form, in addition to associating form with form and meaning with meaning. Thus the network will be able to perform the task of producing a sequence of segments, given a meaning or producing a meaning, given a sequence of segments.

[5]There is a practical problem with this assumption, since I have not really tested whether prediction actually helps with perception. In most cases, phonotactic constraints make it *harder* to perceive words because it means there is greater likelihood that the same subsequences will occur. That is, if words consist of random sequences within those that are phonotactically possible, the more phonotactic constraints, the fewer possible sequences there will be and the more overlap among words. While phonotactics clearly helps in production, learning about phonotactics might not help perception at all, as demonstrated by Gasser (personal communication). One exception might be under conditions of noise: if the input is corrupted, phonotactic constraints might aid in perception because they allow the network to fill in or correct input in places where it is wrong or uncertain.

4.3 Summary

The model employed in the current study is a relatively constrained three-layer network, one in which feed-forward connections are supplemented by limited feedback connections. The input layer consists of three cliques of units: Form, Meaning, and Context. The network's task is that of duplicating the current input and predicting the next form on the output layer.

There are many reasons why the particular architecture shown in this chapter was chosen. The model employed in this thesis is a slightly modified version of the SRN developed by Elman. It has the ability to learn the kind of temporal processing that is a prerequisite to the perception and production of words, by storing the past history on context units. The model was trained on autoassociation as well as prediction. Autoassociation was used to force the network to distinguish the different input patterns on the hidden layer. The task of prediction exactly matches that of production, since the system is trying to decide which phonological forms to produce following the current one. By incorporating both Form and Meaning units, the model has the ability to associate form with meaning as well as meaning with form, making it possible for it to learn to perceive and produce words.

5

Experiments

In this chapter, the experiments performed will be described, including a description of the stimuli used, the training regimen employed, and the three main categories of morphological processes which were learned by the network. Also reported will be extensive analyses of the hidden layer representations and experiments involving knowledge transfer.

5.1 Introduction

In a series of experiments, networks with the architecture described in the Chapter 4 were trained on various morphological forms to test the ability of the model to learn behavior which can be described in terms of a change from an underlying to a more surface form, that is, a process normally thought of as the application of a rule. To answer some of the questions raised earlier in Chapter 2, first, extensive analyses of the hidden layer were done to test the system's ability to encode underlying representations (*URs*), and then the network was tested if it could achieve knowledge

transfer.

The results indicate that the network used is capable of learning morphological rules by encoding URs on the hidden layer and using them in perception and production tasks. That is, given training on the singular, but not the plural of *chip*, the network was later able to generate the appropriate plural suffix following the stem or to determine the grammatical number of /ʧɪps/, a form it had never seen.

For example, the network was trained on pairs like the following:

(5.1) LIP + SINGULAR --> /lɪp/

(5.2) LIP + PLURAL --> /lɪps/

(5.3) CHIP + SINGULAR --> /ʧɪp/

and then it was tested on pairs like the following to see if it then yielded correct morphological forms:

(5.4) CHIP + PLURAL --> /ʧɪp/ + ??,

where the items in capitals represent meanings. In the testing phase, the network was given the appropriate segments for the stem successively, along with the meaning of that stem and the g-number unit indicating plural and predicted the next segment according to its training. The predictive output units were then examined at that point in the cycle where the plural morpheme should normally appear. This is different from a task in which no actual segments are given as input and the network is trained to produce the whole word, stem and suffix altogether:

CHIP + PLURAL --> ??.

For the latter type of task, the inputs are the outputs from the previous time step. This issue will be brought up again later.

However, this solves the problem in only the production direction. The model should also be able to predict meanings, given forms. The model was trained on (5.5), (5.6), and (5.7) and tested on (5.8) to see if it was able to get the correct grammatical number.

(5.5) /lɪp/ --> LIP + SINGULAR

(5.6) /lɪps/ --> LIP + PLURAL

(5.7) /ʧɪp/ --> CHIP + SINGULAR

(5.8) /ʧɪps/ --> CHIP + ??.

5.1.1 Types of Morphological Processes

As shown in Chapter 2, the morphological processes used were classified into three different categories: (1) addition (2) deletion, and (3) mutation, and experiments were conducted on different types of rules; ones which are rarely found in human languages as well as ones which are commonly found, as shown below:

1. suffix (+ assimilation): `fik -> fiks, gob -> gobz`
2. prefix (+ assimilation): `fik -> sfik, gob -> zgob`
3. infix (gemination): `ipa -> ippa`
4. initial deletion: `fik -> ik`
5. medial deletion: `ippa -> ipa`
6. final deletion: `fik -> fi`
7. tone change: `fík -> fik`

8. reversal: `fik -> kif`

9. Pig Latin: `fik -> ikfe`

Types 1 and 2 are common, types 3–7 are less common, type 8 is non-existent, and type 9 is apparently encountered only in language games.[1]

5.1.2 Hypotheses Revisited

Three hypotheses (see Section 2.5) motivated the experiments.

Hypothesis 1 In an appropriate model, a single network should perform both perception and production. Linguistic knowledge is meant to belong to "competence" and should thus be shared by both production and perception.

Hypothesis 2 An appropriate model should be able to learn morphological processes without the benefit of a priori underlying representations or *explicit* rules. Among many rule types, the network should succeed on rule types which are common in human languages and should fail on those which are rare or non-existent. In addition, the UR should be encoded as the hidden layer representation.

Hypothesis 3 The training of the simple recurrent network should capture some of the central generalizations about sound patterns. Specifically, the model should

[1]The frequency of occurrence of different rules as shown here is generally accepted as true, even though there has not been any attempt to record all the known rules in all human languages, classify them according to rule types, and rank them by frequency. Such a task is not feasible, because there are so many different rules across all the world's languages. Hawkins and Cutler (1988) try to explain why suffixation is more frequent across languages than prefixation, and why both are considerably more frequent than infixation. Donegan and Stampe (1979) claim that insertion rules are more frequent than deletion rules. Dinnsen (1979) lists the general arguments against mirror-image rules.

be able to learn rules which are similar to those it was trained on more easily than ones which are very different from those it was trained on.

5.2 Stimuli

Input words were composed of sequences of segments. Since phonological processes do not treat phonemes as atomic, unanalyzable wholes but refer instead to their constituent phonetic properties like voicing, tenseness for vowels, and tongue position, it was necessary that such fine-grained information be present in the network. To meet this end, each segment was represented as a binary vector encoding modified Chomsky-Halle phonetic features (Chomsky and Halle, 1968): 1 for the presence of a particular feature and 0 for its absence. I chose the Chomsky-Halle feature system for the several reasons. First, it is a well-established theory that has sustained rigorous testing over 20 years. Second, I wanted to use a binary feature system rather than a multi-valued one, since the former is much easier to adopt in a connectionist network, such as used in this thesis. Third, I needed to have a small set of features, since redundancy is an important criterion for selecting a feature system. The Chomsky-Halle feature system tends not to be redundant. Redundancy can only make learning easier and might give an unfair advantage to the network when presented with morphological forms.

The distinctive features included in the experiment are shown in Figure 5.1. Each segment type was uniquely specified as a binary vector of 8 or of 13 features depending on the task at hand. More information on feature matrices and input vectors can be found in Appendix A. In order to have as small an input layer as possible, input patterns of 8 feature binary vector were used on most of the experiments, except

1. vocalic
2. consonantal
3. high
4. back
5. low
6. anterior
7. coronal
8. round
9. tense
10. voiced
11. continuant
12. nasal
13. strident

Figure 5.1: Chomsky-Halle Distinctive Feature Matrix.

for the English plural morpheme acquisition task, where input patterns of 13 feature binary vector were employed.

The representation scheme used in this study, that is, the phonetic feature representation, is borrowed from standard generative theory. Feature bundle representations do not in themselves fully represent the systematic nature of phonologic events. Other kinds of representational schemes exist; for example, a more recent theory, often called **non-linear phonology**, departs from the feature matrix type of representation and posits a different kind of representation: a "non-linear" representation, where features are organized in tiers of parallel symbol sequences (cf. Goldsmith, 1990). Of course, it would be even more desirable if the inputs to the system were raw speech data, since then the data would be properly grounded (see Harnad, 1990, for a discussion of the symbol grounding problem). The issue here is whether inputs should be prelabeled segments (or autosegments), as opposed to spectral slices or some other real signal. However, given the scope of the task at hand, the decision was made to utilize a generative-type feature matrix; the use of a feature matrix seems

unproblematic for the current study, and it was assumed that the decision to borrow from standard generative phonology would not bear on the model's performance.

Twenty words were selected for each simulation. I could have used more words on each experiment, but it would have made the tasks more difficult for the model, since the more words used in the simulation, the more rote learning the network would have to perform. Actually, I started with more words for the English plural task and did not get good results until I reduced the data set to 20 words. Note that the generalization results I got are even more impressive given the small number of exemplars for the rules and the large number of testing words. Ten sets of randomly generated artificial words were used for each experiment, except for the English task, which used actual English data. Twelve words from each set were designated "training" words; the remaining eight were "test" words. For each of these basic words, there was an associated inflected form. For convenience, the uninflected form will be referred to as the "singular" and the inflected form as the "plural" of the word in question. The network was trained on both the singular and plural forms of the training words and only on the singular forms of the test words. Therefore there were 32 different input patterns for each simulation. Words were presented one segment at a time. Each word started and ended with a word boundary pattern consisting of all zeroes.

Each "meaning" consisted of an arbitrary pattern, composed of 3 activated units on out of a set of 6 "stem" units, representing the meaning of the "stem" (s-meaning) of one of the 20 input words[2], plus a single unit representing the grammatical number (g-number) of the input word (0 for singular, 1 for plural).

[2]With 6 units used to represent the words where exactly 3 units are turned on at a time, we get $C(6,3) = 20$ possible words.

5.3 Training Regimen

Ten separate simulations were performed for each of the morphological rules, except for the English plural acquisition task, in which 2 runs were conducted.[3] Pilot studies were performed to estimate the optimum size of the hidden layer.

The meaning inputs and the target meaning outputs were constant throughout the presentation of a word, except in the cases where *uncertainty* was involved as explained below. The network was trained on autoassociation for both forms of inputs, that is, both segments and meanings, as well on prediction for the next segment. To test the network in the perception task, that is, the prediction of meaning, given form, the Form units on the input layer were given the sequence of segments in the word, and the g-number unit was set initially to an intermediate value of 0.5. As the word was presented, the output g-number unit should begin to take on the correct activation. In the production task, the Meaning units on input layer were set to the appropriate values, and the **prediction** Form units on the output layer were examined after each input segment. The Form units on the input layer were given the appropriate input segments.

Initially the network was trained on *basic* autoassociation and a prediction task: the network was always given noise-free segments and meanings as input; the desired outputs were, the same segment, the token's meaning, as well as the next segment. This resulted in the desired learning in the prediction direction, but not in the perception direction: the network could predict the correct plural morpheme given the stem and the meaning, but it failed to produce the correct g-number given phonetic

[3]It is not unusual to have variation in the results on different runs; it has been shown that the back propagation learning algorithm is sensitive to initial weights (Kolen and Pollack, 1990a, 1990b). This is one of the reasons why more than one simulations were run.

Table 5.1: Inputs for the word *chip* for the 'unknown number' case.

Segment	Stem Meaning	Grammatical Number	
ʧ	CHIP	0.5	(target=0)
I	CHIP	previous NUMBER output	(target=0)
p	CHIP	previous NUMBER output	(target=0)

form of the word and its s-meaning. To help the network learn in this direction, *uncertainty* was introduced in the input and the network was retrained: on 4 out of every 5 words, the network was given complete words and meanings as before. On 1 out of every 5 words, the input g-number was treated as unknown. That is, the g-number unit was set to an intermediate value of 0.5 initially and for the subsequent input segments to the value that it took on the previous time step. For example, the inputs for the word *chip* for the 'unknown number' case are given in Table 5.1.

Chauvin (1988) also found it necessary to adjust the training regimen for his somewhat similar study.

> The training consists in autoassociating the 28 pairs patterns-labels using the following method. Each cycle (pattern presentation) consists of three steps: at step 1, only the label is presented to *S-in* and the error is computed and back-propagated from *S-out* only; at step 2, only the corresponding image is presented to *W-in* and the error is computed and back-propagated from *W-out* only; at step 3, both label and image are presented to *S-in* and *W-in*, and the error is computed and back-propagated from *S-out* and *W-out*. The complete cycle is therefore composed of three autoassociations: one for the labels only, one for the images, and one for both together. (p.47)

Why did training with uncertainty improve the network's performance? First, the noise that resulted from having the g-number unit copy its activation value from the output layer effectively expanded the data set. Even though the network took longer to learn, the input with noise ultimately helped the network perceive better, given novel words. Second, the introduction of noise helped the network defer its decision on grammatical number until it saw conclusive input data. Third, this training procedure was closer to the actual testing procedure, and it is assured that the trajectory of the weight space moved along the "desired" paths.

Two different strategies were used to update the weights. In the case of *uncertainty* the weights were changed only after a complete set of words was presented in order to avoid the dangerous possibility of getting stuck in a local minimum. Since a part of the output was copied back to the input layer, it was suspected that a few atypical guesses could pull the network into some part of the weight space where it might become trapped.[4] In other cases, *online* updating was used to save memory space and to allow the network to learn the task in fewer epochs[5] (Fahlman, 1988).

To accelerate the training time *adaptive training* (Allen, 1989; Allen, 1990; Elman, 1991) was used. The learning rate began at 0.25 and was decreased by a factor of 0.75 when there was no improvement over a period of 5 epochs. A fixed momentum of 0.9 was used. Elman (1989a) reports a momentum of 0 for some of his experiments, but for the current experments a network with zero momentum would have taken too

[4] Kushner et al. (1991) report a relevant finding in their experiments on tasks involving sequences. The task was prediction, where the 6th element in a sequence depended on what the 2nd and 4th elements were. They tried this two different ways. One network was just trained to predict the next element each time. The other was trained to predict only the 6th element; there was no training at all as the other elements were presented. The second network did better at learning how to predict the 6th element. The prediction training was in effect undoing the differences built into the initial random weights.

[5] An epoch is the training cycle, containing a single presentation of each of the training patterns, one after another.

long to converge, for example, in one case a momentum value of 0.9 gave convergence after 180 epochs, while the network with zero momentum did not converge even after 6250 epochs.

To test the network's performance on the production task, the network was given the appropriate segments for the stem successively, along with the meaning of that stem and the g-number unit indicating plural. Then the prediction output units were examined at the point where the plural morpheme should appear. Using Euclidian distances, each output pattern was converted to the nearest phoneme. This phoneme was then compared to the desired phoneme.

To test perception performance, the network was given the sequence of input segments for a word, the stem meaning units were set to the appropriate pattern, and the g-number unit was given the initial value of 0.5. At the presentation of each new segment, the g-number unit was copied from the output on the previous time step. The output g-number unit was examined after the appearance of either the appropriate plural form or the word boundary.

5.4 Experiment 0: Simple Plural Cases

Before the network as described in Chapter 4 was used for the real tasks described above, a modified network was employed to find out whether this kind of simple recurrent network is capable of and appropriate for performing the kind of task involving (1) production and perception and (2) the association of form with meaning.

5.4.1 Method

CVC words from an artificial language were used to test the "suffix" rules which added an /s/ or /z/ to singular words to form plural words. The suffixes agreed on the **voice** feature with the previous segment. For this particular simulation, the input/output layer did not have meaning units, but only a grammatical number unit. This was done to create as small a network as possible. There were 9 units on the input layer; 13 on the hidden layer; and 17 on the output layer. The network was trained on 120 words. For 20 of the words, singular and plural forms were presented while training. The remaining 100 words were presented only with singular forms. Another set of 100 words were reserved for testing. There were no duplicates among the 220 words. Learning was stopped after 50000 word presentations (about 350 epochs).

5.4.2 Results

Since no meaning units were present in the network, it could not naturally produce correct words; yet it was able to predict the endings correctly. The network learned the training set perfectly, that is, all endings (either # (word boundary), /s/, or /z/) were correct. When the test words were presented, it also predicted the endings correctly, too. When the network was given 100 more test words which had never been seen, it still could predict all of the endings correctly. Even though the training set did not have any words which ended /g/ or /f/, the network predicted the endings correctly for all of the test words which ended /g/ or /f/. The network produced a word boundary after the third phoneme, if the word was singular; /s/ or /z/, if the word was plural. The network also learned to autoassociate phonetic forms.

On the perception task, the network predicted singular after the third phoneme as a default number. If the next segment was a word boundary, it did not change. Once /s/ or /z/ appeared, then it correctly changed the number to plural.

To see what was happening, after the third phoneme was input, hidden unit activations were collected and a hierarchical clustering analysis[6] was performed. The dendrogram for the cluster analysis is shown in Figure 5.2. For singular words, groupings were shown according to the phonetic features *place* and *stop*, for example, stop vs. fricative. For plural words, words were clustered according to the *voicing* feature.

As shown in Figure 5.3, a box plot[7] of hidden layer activations after the third segment shows that each unit encoded a particular feature. Unit 1 is dedicated to voiced consonants; unit 5 represented alveolar stops (/d/ or /t/); unit 7, velar stops (/g/ or /k/); units 8 and 10, singular and plural words, respectively; unit 12, fricatives (/v/ or /f/); and finally, unit 13, bilabial stops (/b/ or /p/). These results are not surprising given the task of autoassociation and are exactly what was expected from the model.

Note that even though this task only involved predicting grammatical number, the context units were needed here. Without them, the network could not have distinguished between two identical phonemes in the first and the third position, for example, two /b/'s in *bob*, and it might have been impossible for the network to decide when to add a plural morpheme.

[6]This is a statistical technique that operates on a distance matrix of all pair-wise distances between objects. When plotted using a dendrogram, the two groups of leaves closer together are seen as most similar in characteristics.

[7]This plot, sometimes called a *Hinton diagram* after the person who first used this type of plot to visually display weights in a connectionist network, is a way of showing relative strength of each unit's activation. Big boxes represent activation values close to 1.0 and dots denote values close to 0.0.

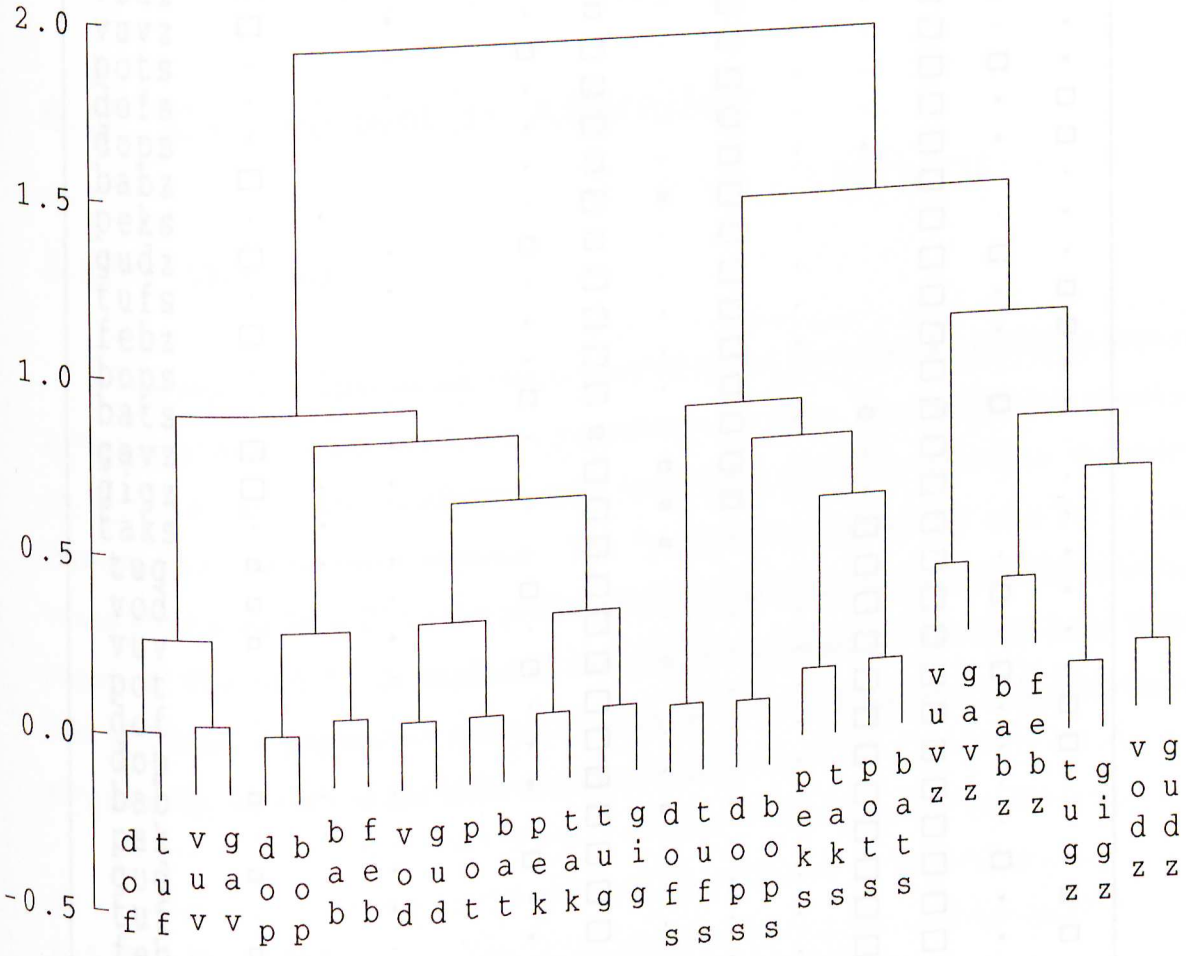

Figure 5.2: Hierarchical clustering of some of the words used in the experiment, according to their hidden unit activations in the simple plural task.

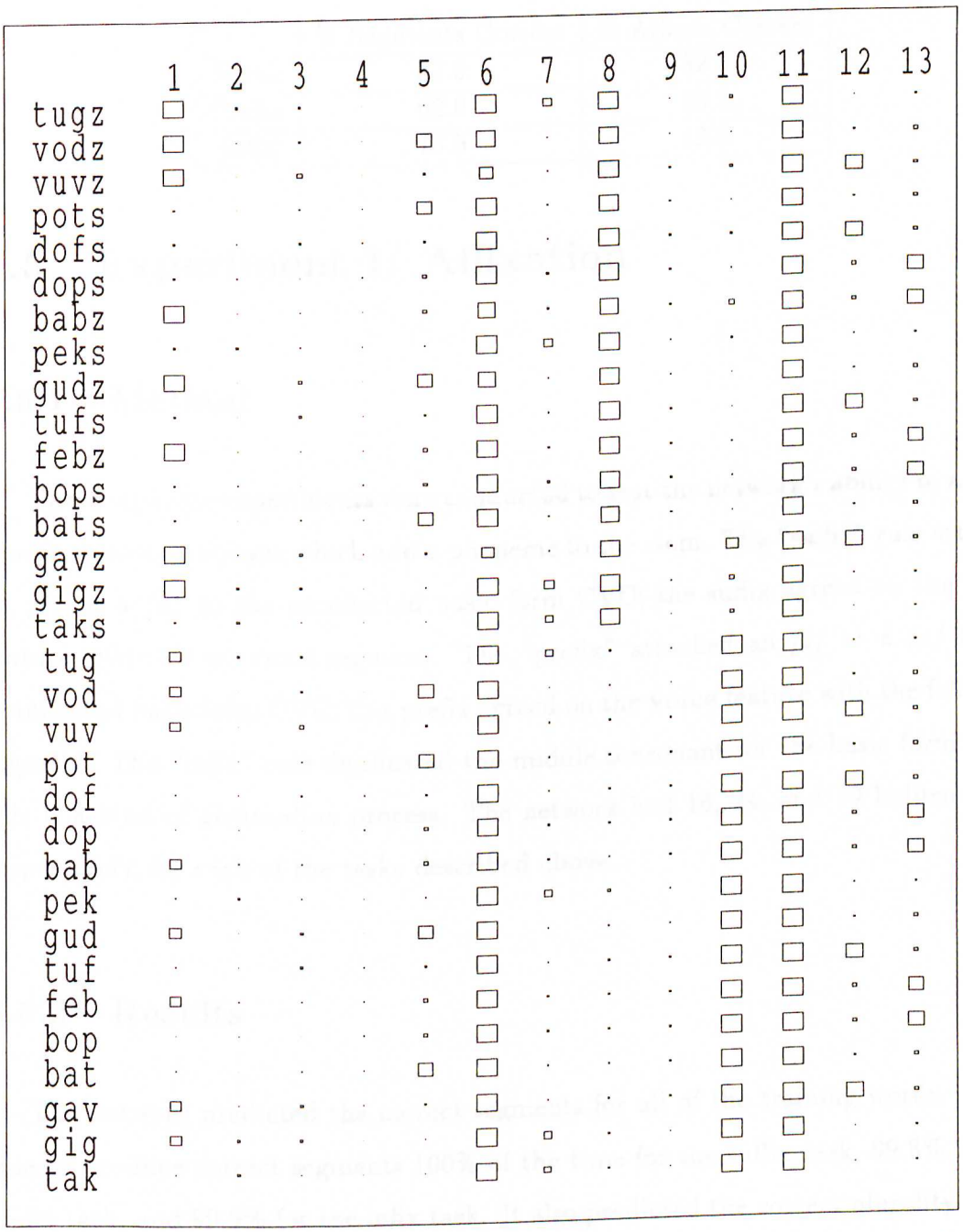

Figure 5.3: Box plot of hidden layer activations after the third segment was input in the simple plural task.

Table 5.2: Results for the production of test words in the affixation experiments.

	% Segments Correct	% Affixes Correct
Suffix	82.3	82.5
Prefix	62.0	76.3
Infix	73.5	42.5

5.5 Experiment 1: Affixation

5.5.1 Method

Three separate experiments were conducted to test the network's ability to acquire morphological processes which add a phoneme to the stem. The "suffix" rule attached an /s/ or a /z/ to the uninflected basic form CVC; the suffix agreed on the **voice** feature with the previous segment. The "prefix" attached an /s/ or a /z/ to the uninflected basic form CVC; this prefix agreed on the **voice** feature with the following segment. The "infix" rule duplicated the middle consonant for the basic form VCV; this is a kind of *gemination* process. The network had 16, 24, and 19 hidden units, respectively, for each of the tasks described above.

5.5.2 Results

The network predicted the correct segments for all of the training words. It was able to produce correct segments 100% of the time for the suffix task, 99.8% for the prefix task, and 99.9% for the infix task. It also predicted the correct plurality 96.5% of the time for the suffix task, 94.5% for the prefix task, and 98.4% for the infix task.

When test words were presented, most of the test forms were correctly predicted

Table 5.3: Results for the perception of test words in the affixation experiments.

	% Plurality Correct
Suffix	79.0
Prefix	76.0
Infix	90.0

for the "suffix" and "prefix" rules, while less than half were correctly predicted for the "infix" rule.

Results for the production of test words are summarized in Table 5.2. The network was very good at generating the suffixed forms, a little less good at the prefixed forms, and less than average for overall segments correct at the infixed forms. Note that chance for these tasks is 18.8%, 18.8%, and 37.5%, respectively.

Table 5.3 shows the results for the perception of plurality on the plural forms of the test words. Note that chance is only 37.5%.[8] On this test, infixed forms performed better than the prefixed or suffixed words. This is because the relevant information (the second consonant) appeared at just the right place. In the case of prefixation, the critical information appears at the beginning of the word, forcing the network to make a decision based only on this information. As a consequence, there still remains ambiguity: is the first phoneme /s/ a prefix, or is it a part of the stem? In the suffixation case, the critical information appears too late, meaning that a wrong path to the final phoneme will likely give a wrong answer.

[8]Out of 32 training input words, only 12 are plural. This percentage is same for all the perception tasks reported in this chapter.

5.6 Learning the English Plural "Rule"

Since the previous "suffix" rule experiment was so successful, an experiment on the acquisition of the English plural "rule" was conducted to test it on the real data. The English regular plural morpheme has three variations: /s/, /z/, and /ɪz/. To account for them in generative phonology, two rules are invoked as explained in Section 2.4: voicing assimilation and vowel insertion.

5.6.1 Method

The overall structure of the network is shown in Figure 4.1. The Form clique in the input and output layers consist of 13 units, which represent a single phonetic segment. The Meaning clique has 7 units; 6 for the "stem" and 1 for the plural marker. The network has 32 hidden units and also 32 Context units.

The input corpus for this simulation consisted of a set of 20 one-syllable English nouns as shown in Table 5.4.

Twelve of these (*top, lip, ache, hat, Tom, saw, job, zone, judge, dish, page, catch*) were designated "training" words, while the other eight (*chip, bat, cake, mom, tub, tone, match, age*) were "test" words. Note that both the training and the test words include nouns taking all three of the plural morpheme allomorphs. The rest of the training regimen was exactly that described in the beginning of this chapter. Two separate runs were conducted.

Training continued until the network performed correctly on the training set, that is, until the error for every output was less than 0.05: 270 epochs for the first run;

Table 5.4: List of words and their representations used in the English plural acquisition task.

Words	ASCII Transcript	IPA Transcript	Training	Testing
top	tap	tɑp	x	
lip	lIp	lɪp	x	
ache	ek	eᵻk	x	
hat	hAt	hæt	x	
Tom	tam	tɑm	x	
saw	s)	sɔ	x	
job	j^b^9	ʤɑb	x	
zone	zon	zon	x	
judge	j^j	ʤʌʤ	x	
dish	dIS	dɪʃ	x	
page	pej	peᵻʤ	x	
catch	kAC	kæʧ	x	
chip	CIp	ʧɪp		x
bat	bAt	bæt		x
cake	kek	keᵻk		x
mom	mam	mɑm		x
tub	t^b	tʌb		x
tone	ton	ton		x
match	mAC	mæʧ		x
age	ej	eᵻʤ		x

Table 5.5: Results of English plural acquisition experiment.

	Production		Perception		n
	Sg	Pl	Sg	Pl	
Training	100%	100%	100%	100%	24
Testing	100%	87.5%	100%	100%	16

160 for the second.

5.6.2 Results

The results of English plural acquisition experiment are summarized in Table 5.5.

The network predicted the correct segments for all but 2 of the stem segments and 14 of the 16 suffixes in the two runs. The two suffix errors involved substituting /ʧ/ for /s/ in *bats* and /f/ for /z/ in *tones*.

For the training words, the output g-number unit fluctuated around 0.5 until the relevant information was given. Then the unit correctly turned on or off according to whether the word boundary or plural ending appeared. For the test words, the network consistently output 0 before the appearance of the relevant information. This is not surprising since the network only saw singular forms of these words during training. When the word boundary appeared in the input, the unit correctly stayed off. When the plural morpheme appeared in the perception task, the output number unit behaved appropriately for all of the 16 test items.

[9]This should be *jab*; yet it is safe to assume that it would not have affected the results of the experiments. Later experiments using the correct transcription *jab* show almost identical results.

5.7 Experiment 2: Deletion

5.7.1 Method

In this set of experiments, one of three rules was used to generate plural forms in which a segment was deleted from some part of the singular form of the word: from the beginning of a word (CVC → VC) ("pre-del"), from the middle (VCCV → VCV) ("mid-del"), and from the end (CVC → CV) ("post-del"). There were 10 separate simulations for each of the three morphological rules. After the pilot run the size of hidden layer was decided upon: for pre-del the hidden layer had 26 units; for mid-del 20; and for post-del 26.

5.7.2 Results

The network learned the set of training words for all three rules quite successfully. Segments were produced correctly more than 99% of the time (pre-del: 100%; mid-del: 99.6%; post-del: 99.9%), and the network correctly predicted grammatical number more than 96% of the time (pre-del: 99.1%; mid-del: 96.2%; post-del: 98.4%).

The results for the test words are shown in Table 5.6. The network correctly predicted the word boundary more than half of the time when it was exposed to the test words after being trained to delete the final consonant ("post-del"). It performed quite poorly when a segment was deleted from the middle ("mid-del"), and even worse for the case a segment was deleted from the beginning of the word ("pre-del"). Note that the network failed to learn exactly those types of rules which are rarely found in human languages, however, some aspects of the rules had been learned. Thus in

Table 5.6: Results of deletion experiments. "% Segments Correct" refers to the percentage of the segments which the network predicted correctly after a segment was deleted.

	Production	Perception	n
	% Segments Correct	% Plurality Correct	
Pre-del	12.5	60.0	80
Mid-del	23.8	73.8	80
Post-del	57.5	67.5	80

about 80% of the cases the network produced the correct syllable structure (77% VC for "pre-del", 80% CVC for "mid-del"; in "post-del" cases it always predicted CV). The network performed little better than chance on the perception task.

5.8 Experiment 3: Mutation

5.8.1 Method

Three different kinds of experiments were performed to test the mutation rules. In one experiment, the network was trained to change a single feature of all of the segments in the singular word to generate the plural word. This is analogous to a tone change rule where singular words have all low tones, and plural tones all high tones ("H-L Tones"). In another experiment, the "reversal" rule was tested: plural words were generated by reversing the segments of the singular words. The third experiment involved *Pig Latin*, where the word-initial consonant of the English word is moved to the end and the vowel [e] is added after it. As with the previous experiments 10 separate simulations were run. The H-L Tones network performed best with 20 hidden units; The Pig Latin network 17; and the reversal network 21.

Table 5.7: Results of mutation experiments. "% Segments Correct" refers to the percentage of all the segments which the network predicted correctly.

	Production	Perception	n
	% Segments Correct	% Plurality Correct	
H-L Tones	97.5	99.1	80
Reversal	22.5	13.0	80
Pig Latin	27.2	61.3	80

5.8.2 Results

Results are summarized in Table 5.7.

The network was able to learn the "H-L Tones" task (100% correct for both production and perception tasks on training words) very well. It was also very good at generating "high" tones for novel plural words and perceiving a novel word with a "high" tone as a plural word.

Yet the network failed to generate a new reversed form, even though it learned the training words more than 99% of the time. On the perception task, the network performed at a level considerably worse than chance (37.5% as shown in Section 5.5.2). This is apparently due to the fact that during training the network was exposed to the singular and plural forms of training words but only to the singular forms of test words. Thus it saw more singular forms overall and, given no evidence one way or the other, responded with an activation less than 0.5 on the g-number unit. The network found it much harder than other types of rules to learn a reversal, a type of rule which is apparently difficult for human language learners.

The network apparently had difficulty in coming up with the correct Pig Latin forms when given test words, even though it performed well on training words (more

than 95% correct). This is not surprising given the twofold task of rearrangement of segment order and affixation.

5.9 Analysis of Hidden Layers

So far I have reported that the network was able to learn apparent rule-governed morphological processes in a manner that makes use of the associations of form with meaning. The next question I should be able to answer is: where are the URs? It is the pattern on the hidden layer that mediates the relation between form and meaning; in this case, these patterns would be analogous to URs in the networks. Only analyses of *plurals* are reported here. Since *teacher forcing*[10] was employed in training, the hidden representation of the stem for a given word is naturally similar during the perception and the production tasks; thus, it is not possible to draw a conclusion about URs from the hidden layer representations of the stems.

The question that needs to be asked is: Is there a single hidden-layer pattern shared by different realizations of the plural morpheme? As will be shown, an analysis of the hidden layer representations of the network indicates that certain units are dedicated to representing the plural morpheme, independent of its surface form. Thus it appears that my networks have the capacity to learn distributed URs.

[10]Throughout the experiments, "correct" inputs were given, the activation values were propagated to the output units, "correct" outputs were compared to desired outputs, weights were adjusted, new actual inputs were given, and the cycle repeated. This is different from techniques in which the actual outputs are used as the inputs for the next time step.

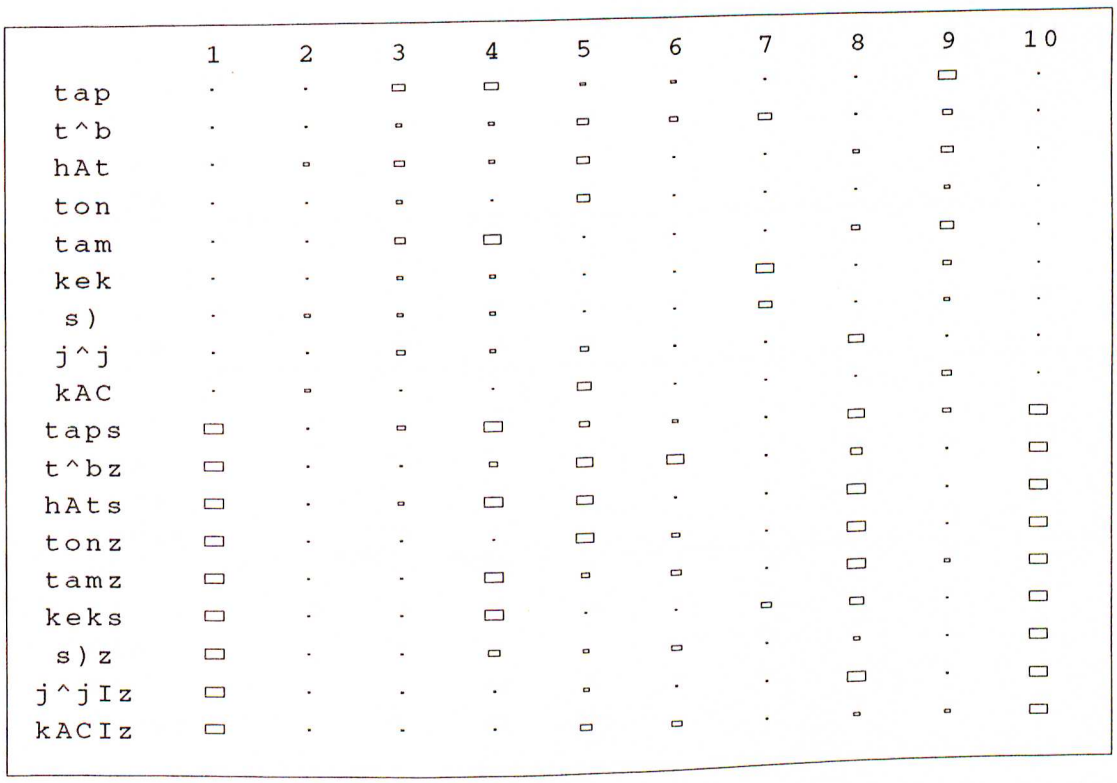

Figure 5.4: Box plot of hidden layer activations before perceiving plurality during the perception task in the first English-plural run.

5.9.1 English Plural Task

Initially, I analyzed hidden unit representations which the network developed during the first run of the English-plural task.

Figure 5.4 shows a box plot of the hidden unit activations for 18 randomly-picked words (test words as well as training words) at the segment before plurality is perceived (perception task): after the appropriate plural form was input for the plural or after the word boundary for singular words. Only 10 selected units are shown in the figure. From the figure it appears that unit 1 and unit 10 distinguish plurality. But further analysis shows that unit 1 always responds to the sibilant phonemes, such as /s/, /z/, and /ʤ/ regardless of their positions in the word (see Figure 5.5).

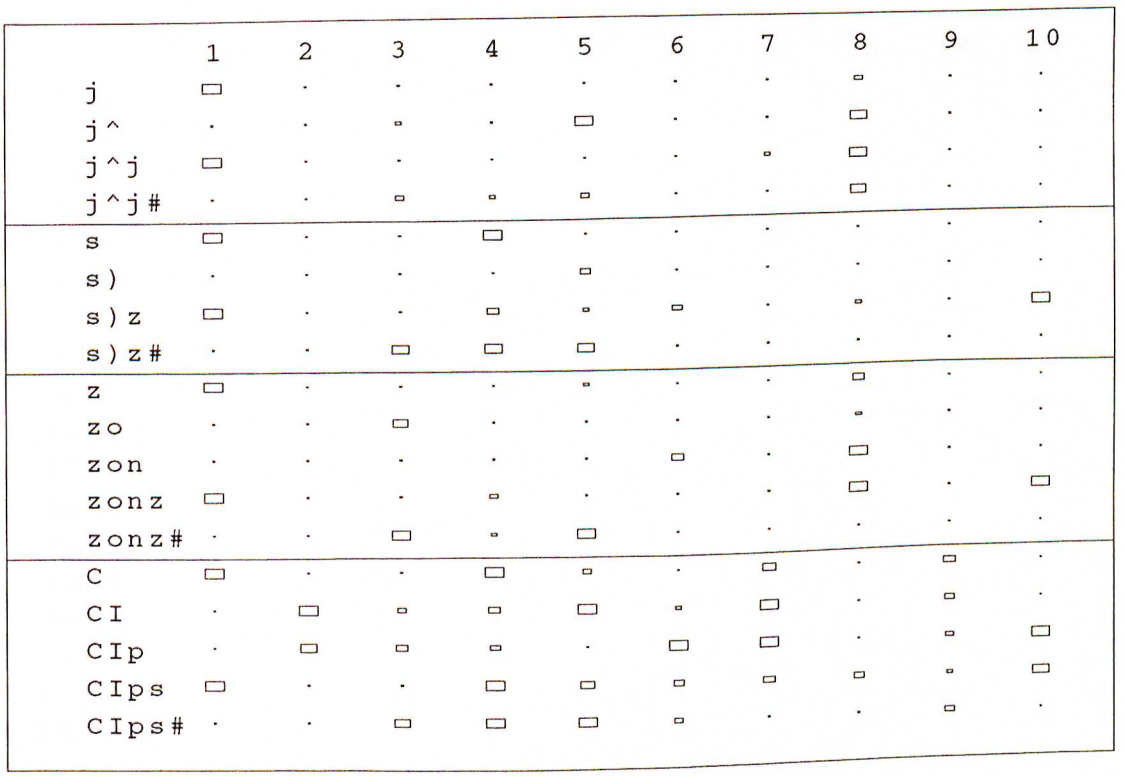

Figure 5.5: Box plot of hidden layer activations for some of the words showing that unit 1 is responsive to the sibilant phoneme and unit 10 encodes plurality during the perception task in the first English-plural run.

Figure 5.5 shows the hidden unit activations for the words *judge, saws, zones* and *chips* after each segment is presented during the perception task. In the figure # denotes a word boundary. Let us examine unit 10 more closely. For *judge* this unit is always off, yet for *saws* and *zones*, it correctly turns on after /z/ is input, signaling a plural form. For *chips* it turns on after the /p/ is presented, making a spurious prediction (and quickly turns off when a word boundary is encountered, which is not shown in the figure) and stays on when /s/ is given at the next time step. In a word, unit 10 is not generally turned on until it has enough information to decide the number of the word in question.

Now what happens when the network is given a production task? In this case I

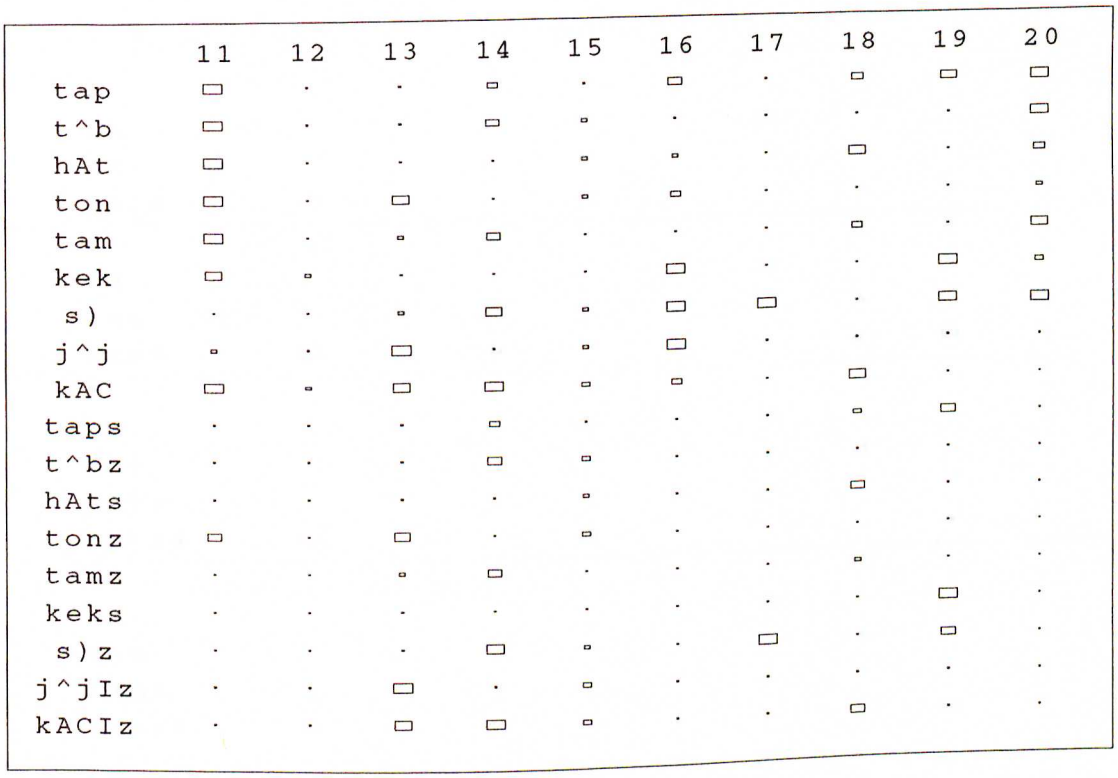

Figure 5.6: Box plot of hidden layer activations after the stems are input during the production task in the first English-plural run.

found that unit 11 is responsible for producing plural morphemes; this unit goes on for the singular words and off for the plural words.

Figure 5.6 shows a plot of hidden unit activations for the production task, that is, after the stems were presented. In the figure 10 different units are shown selected from the same 32 hidden units. Unit 11 for *saw* is off. Even though it correctly predicts the word boundary with the current threshold of 0.5, if we were to use the lower threshold, it predicts /z/, which is the plural for the word. The activation value of unit 11 for *judge* is very small. Again it is close to predicting /ɪ/, which would come next if it were a plural. The activation of unit 11 for *tons* is high, predicting /f/ instead of /z/. One might argue that since number is presented in the input layer

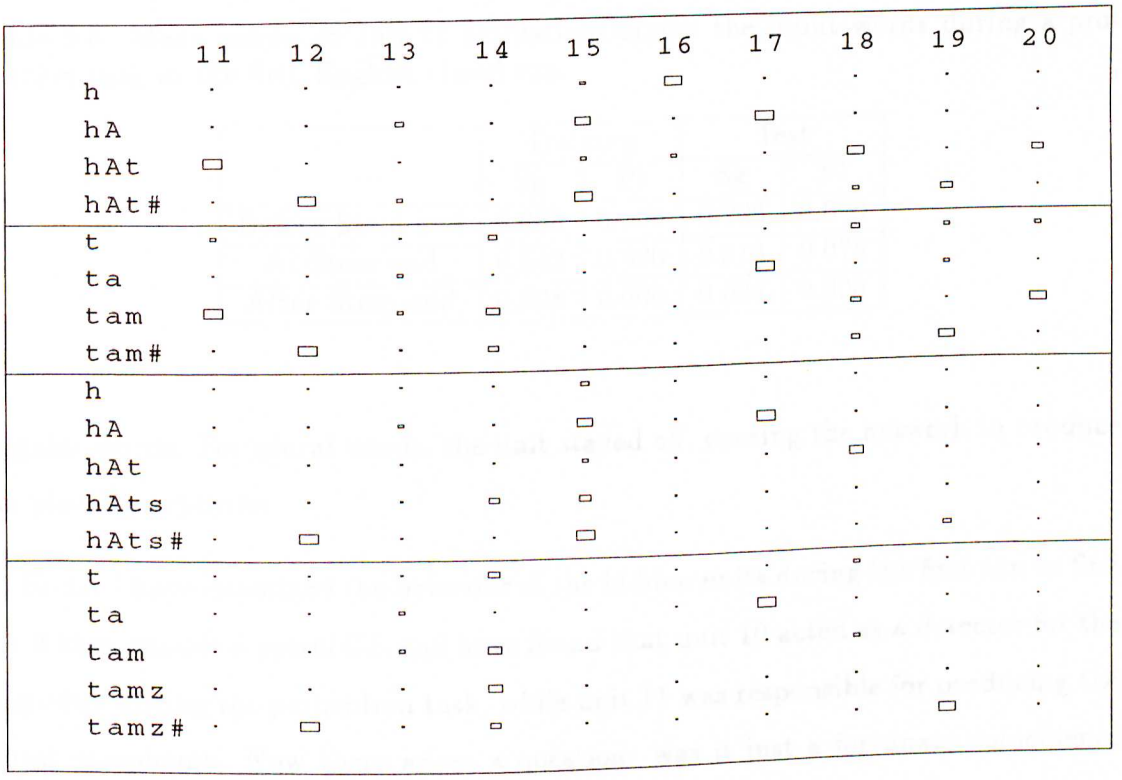

Figure 5.7: Box plot of hidden layer activations for *hat* and *Tom* after each segment is presented during a production task in the first English-plural run (including word boundary).

the unit can easily represent the correct number by simply copying the input value. But this is not the case. The unit turns on only after the stem for singular words.

Figure 5.7 shows the hidden unit activations for the words *hat* and *Tom* after each segment is presented during the production task. If unit 11 were simply copying the value of the number unit from the input layer, this unit should be on all the time for the singular words. It is not. It is turned on only after the stem where the singular should come. Table 5.8 shows the mean values of unit 11's activations across all the input words at "before-stem-end", "at-stem-end', and "after-stem-end" in the word. The table clearly shows that after the stems were presented in the input, unit 11 turned on, causing the word boundary symbol to be produced in the output for the

Table 5.8: Mean values of unit 11 activations across the input words during a production task in the first English-plural run.

	Training		Test	
	Sg	Pl	Sg	Pl
Before Stem-end	0.033	0.000	0.206	0.000
At Stem-end	0.642	0.000	0.813	0.075
After Stem-end	0.008	0.000	0.038	0.000

singular words. For plural words, the unit stayed off, causing the network to produce the plural morpheme.

So far I have examined the behavior of the hidden units during the first run to find out if they encode a *plural* UR and have found that unit 10 acted as a detector for the g-number during the perception task, while unit 11 was responsible for producing the plural morpheme. Now there arises a question: was it just a fortunate coincidence that I was able to find some units which encode plurality? Or is it really safe to say that my network developed a UR? To answer these questions I analyzed the another run.

In the second run it turned out that unit 6 acted as a detector for the g-number for both perception and production tasks. Figure 5.8 shows a box plot for 18 randomly-picked words during the perception task in the second run. Only the first 10 units are shown. From the figure we can easily see that unit 6 might be responsible for encoding g-number. The activation value of unit 6 for *chips* is very small, around 0.2. This is a word for which the network made a wrong prediction: it predicted singular. We can safely say that the activation value of unit 6 is consistent with the results of the prediction task. Looking at Figure 5.8, we might say that unit 10 is also sensitive to the g-number. But further analysis shows that this is not the case. In Figure 5.9 we can see that unit 10f is on for some singular as well as plural nouns.

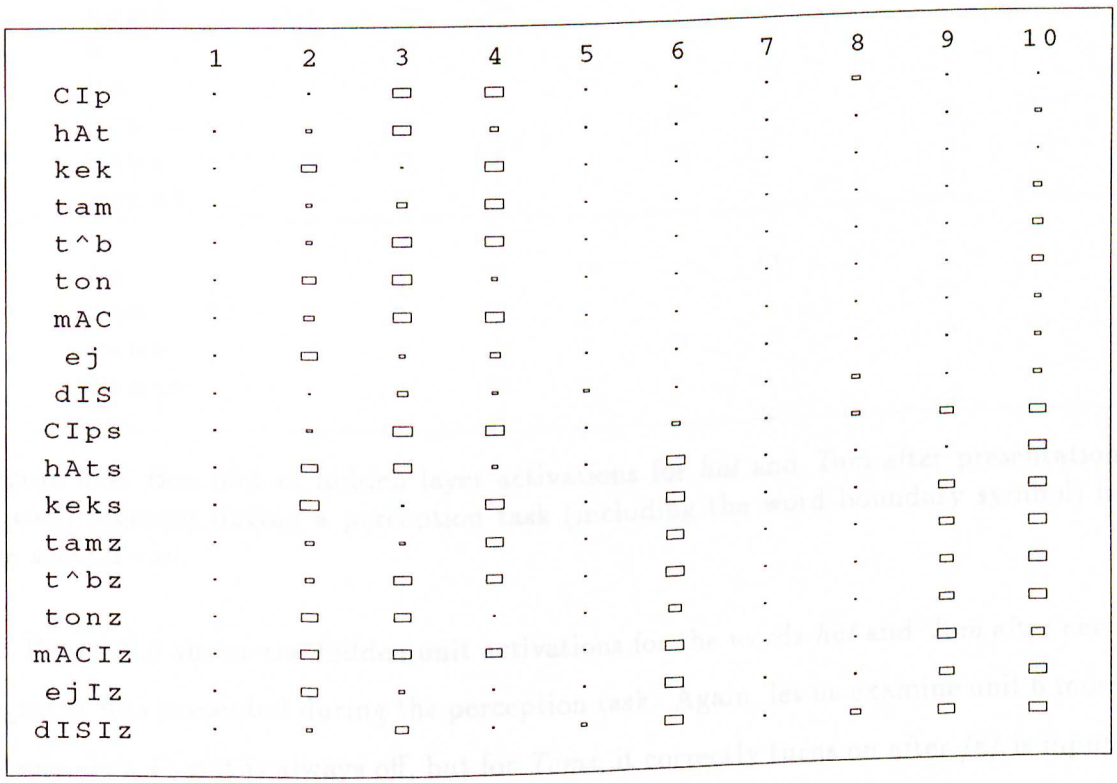

Figure 5.8: Box plot of hidden layer activations before perceiving plurality during the perception task in the second English-plural run.

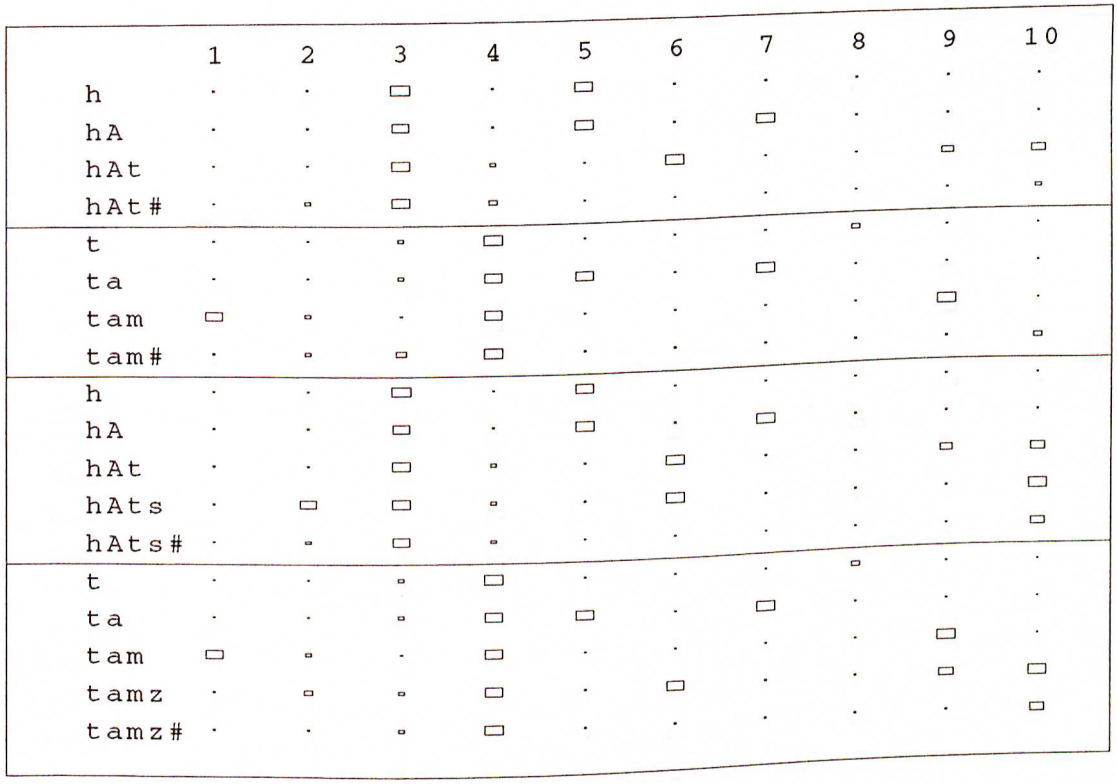

Figure 5.9: Box plot of hidden layer activations for *hat* and *Tom* after presentation of each segment during a perception task (including the word boundary symbol) in the second run.

Figure 5.9 shows the hidden unit activations for the words *hat* and *Tom* after each segment was presented during the perception task. Again, let us examine unit 6 more closely. For *Tom* it is always off, but for *Toms*, it correctly turns on after /z/ is input, signaling that a plural. For *hat* it turns on after /t/ is presented, making a *false prediction*, and quickly turns off when additional information (the word boundary) is available. For *hats* it correctly predicts that the word is plural.

Now what happens when the network is given a production task? Figure 5.10 shows another plot of hidden unit activations for the production task, that is, after the stems were presented. Again unit 6 predicts whether the next segment will be a plural or a singular. As in the case of the first run, the unit changes its value only

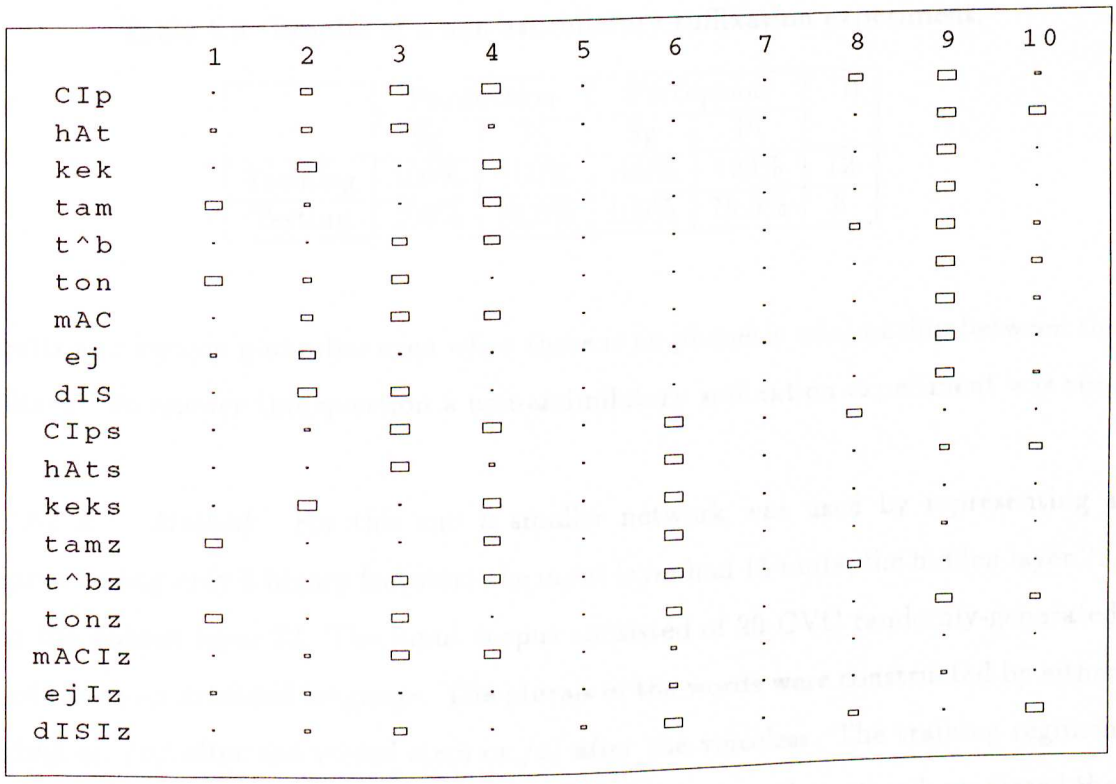

Figure 5.10: Box plot of hidden layer activations after the stems are input during the production task in the second English-plural run.

after the stem is completely presented. For the cases of *matches* and *ages*, the values are small, 0.1 and 0.2 respectively. This seems to be due to the insertion of the vowel /ɪ/.

5.9.2 Non-assimilatory Suffixation Task

After the English-plural runs, I asked this question: does the network really encode the notion of "plurality", or does it just encode the "rule" that adds a sibilant, /s/ or /z/, at the end of nouns, taking advantage of the phonetic cue that points to the right suffix by way of assimilation? In other words, can the network produce correct

Table 5.9: Results of a non-assimilatory suffixation experiment.

	Production		Perception		n
	Sg	Pl	Sg	Pl	
Training	100%	100%	100%	100%	12
Testing	100%	75.0%	100%	75.5%	8

results and encode g-number even when there is no phonetic relationship between the suffixes? To answer this question a non-assimilatory suffixation experiment was run.

5.9.2.1 Method. For this run a smaller network was used by representing a segment using only 8 binary features: the input layer had 15 units, the hidden layer 21, and the output layer 23. The input corpus consisted of 20 CVC randomly-generated words from an artificial language. The plurals of the words were constructed by either adding an /o/ after the voiced stem or /s/ after the voiceless. The training regimen was the same as for the English plural task. After training, the network predicted the correct segments for all of the training words and 6 of the 8 test words. The output g-number unit was correct for all the training words and 6 of the test words. The results are summarized in Table 5.9.

5.9.2.2 Analysis. Earlier the question was raised concerning whether there are any units which are dedicated to representing just the plural morpheme, and not the sibilant phoneme. To answer this question, hidden unit representations which the network developed during the non-assimilatory suffixation run were analyzed.

Figure 5.11 shows a box plot of the hidden unit activations of 18 randomly-picked words at the segment before the network perceives plurality, during the perception task in the non-assimilatory suffixation run. From the figure we can see that unit 3 clearly distinguishes singular words from plural ones.

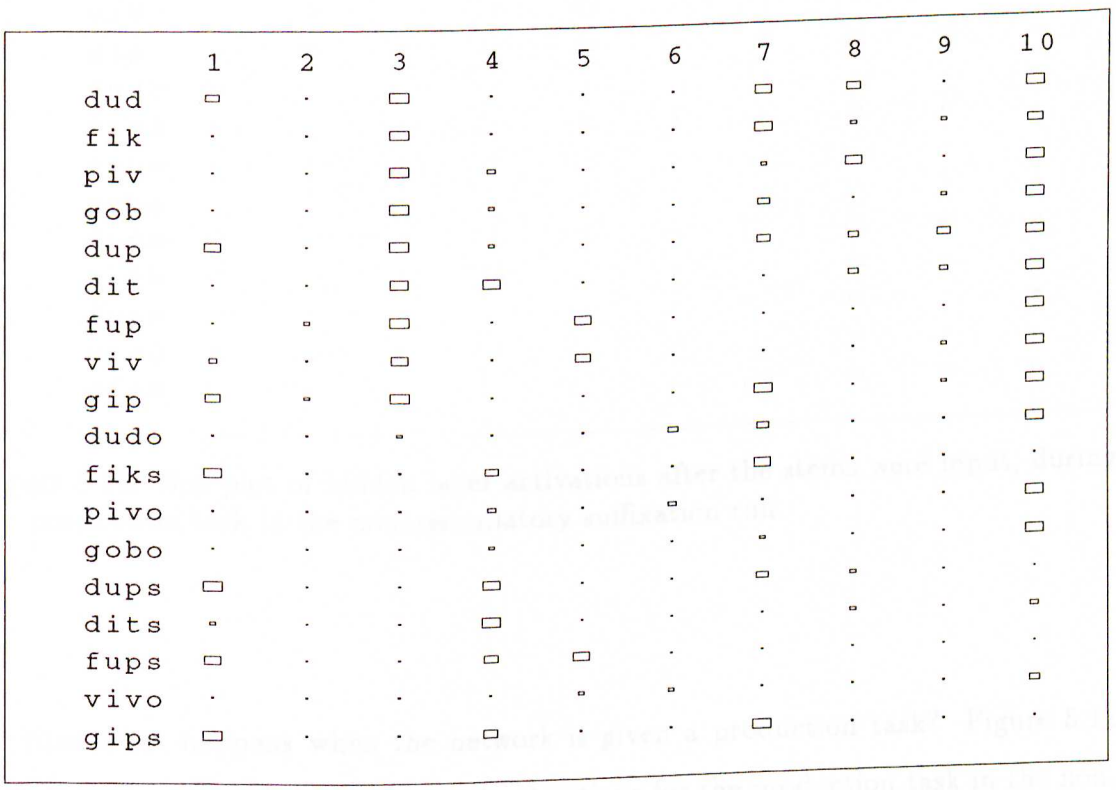

Figure 5.11: Box plot of hidden layer activations before perceiving the plurality during the perception task in the non-assimilatory suffixation run.

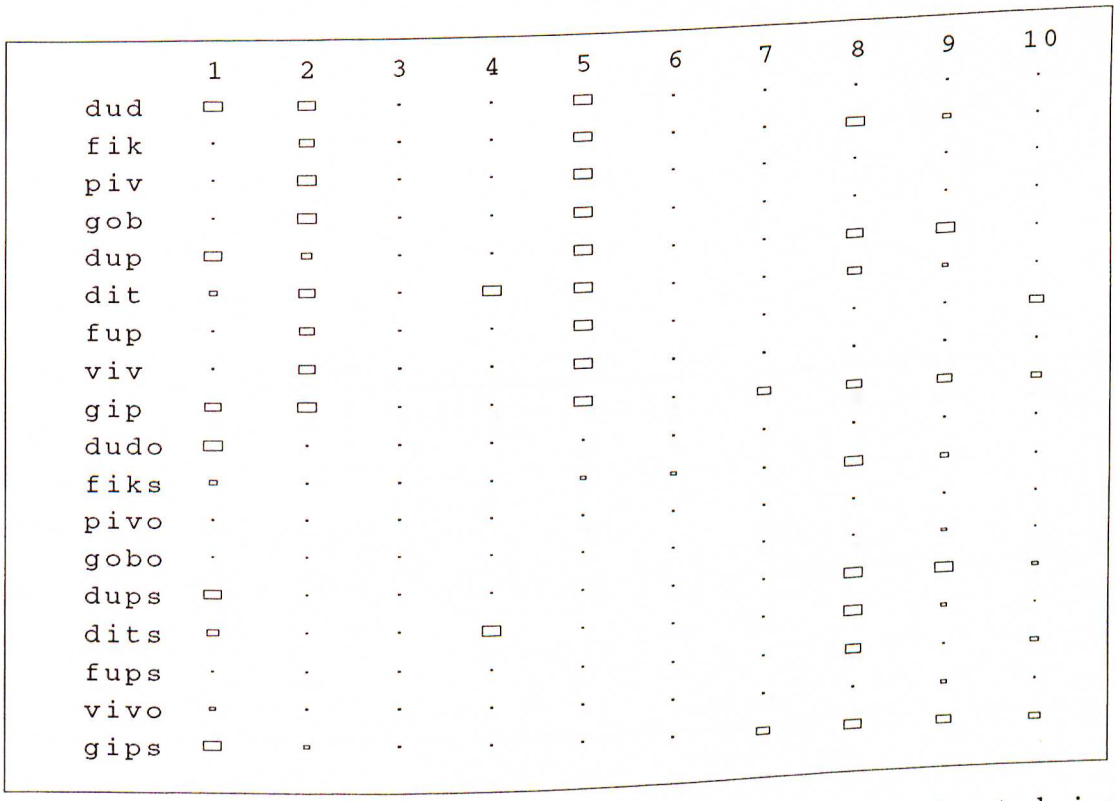

Figure 5.12: Box plot of hidden layer activations after the stems were input, during the production task in the non-assimilatory suffixation run.

Now what happens when the network is given a production task? Figure 5.12 shows another plot of the hidden unit activations for the production task in the non-assimilatory suffixation run. From the figure we can see that the box sizes for unit 2 and unit 5 are quite different for singular nouns and plural nouns. As shown in Figure 5.13 unit 2 turned on only after all the segments of the stems were given and no plural marker was expected, indicating that the next segment should be a singular morpheme. In Figure 5.13 only 4 words are shown. In fact I did the same kind of analysis as shown in Table 5.8 for the production task in the non-assimilatory suffixation run, obtaining the same results.

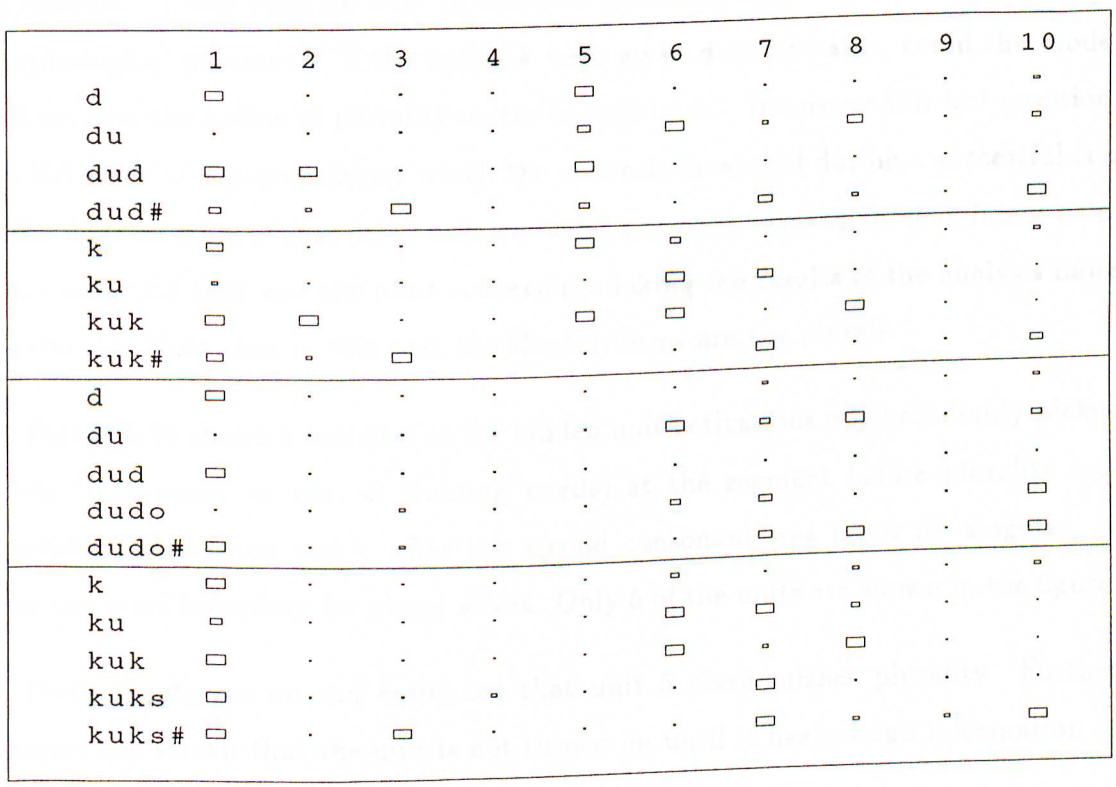

Figure 5.13: Box plot of hidden layer activations for *dud* and *kuk* after each segment presentation step, during the production task in the non-assimilatory suffixation run.

5.9.3 Final Deletion Task

So far, I have analyzed hidden layer representations that were developed during two suffixation task runs. The analyses showed that the network encodes the notion of plurality on the hidden layer as a UR. At this point, I asked myself if this representation was possible only in addition processes. What about other types of morphological processes? If the network were given deletion tasks, could the model still develop the notion of plurality on the hidden layer? To answer this last question, the hidden unit representations which the network developed during a successful run of the final deletion ("post-del") task were also analyzed. Among three different deletion tasks, this task was the most successful, making the results of the analysis more convincing. Note that in this task the shorter forms are the plurals.

Figure 5.14 shows a box plot of the hidden unit activations of 10 randomly-picked words (test words as well as training words) at the segment before plurality was perceived (perception task): after the second consonant was input for singular, or after the word boundary for plural words. Only 5 of the units are shown in the figure.

From the figure we can easily see that unit 5 distinguishes plurality. Further analysis has shown that the unit is not turned on until it has enough information to decide the grammatical number of the word in question.

Now what happens when the network is given a production task? In this case I found that the same unit is responsible for producing plural words; unit 5 is on for the singular words and off for the plural words.

Figure 5.15 shows another plot of hidden unit activations for the production task, that is, after the second segment (vowel) was presented. One might argue that since number is presented in the input layer the unit could easily represent it by simply

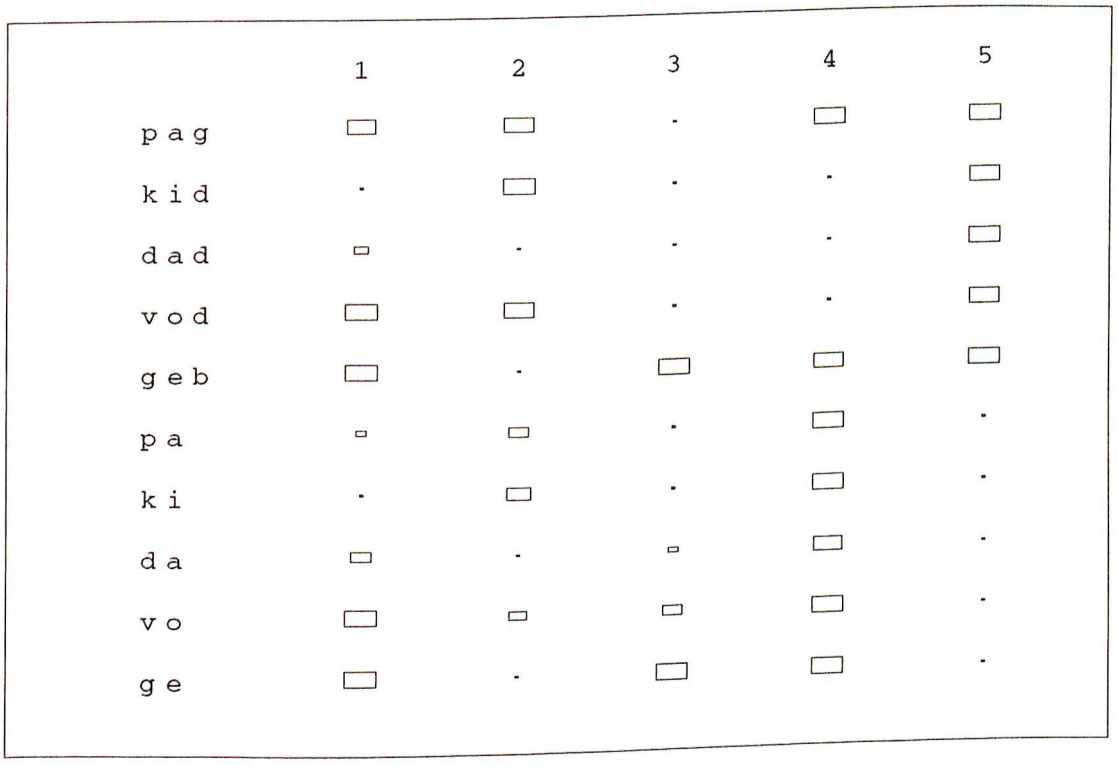

Figure 5.14: Box plot of hidden layer activations before perceiving the plurality during the perception task in a "post-del" task run.

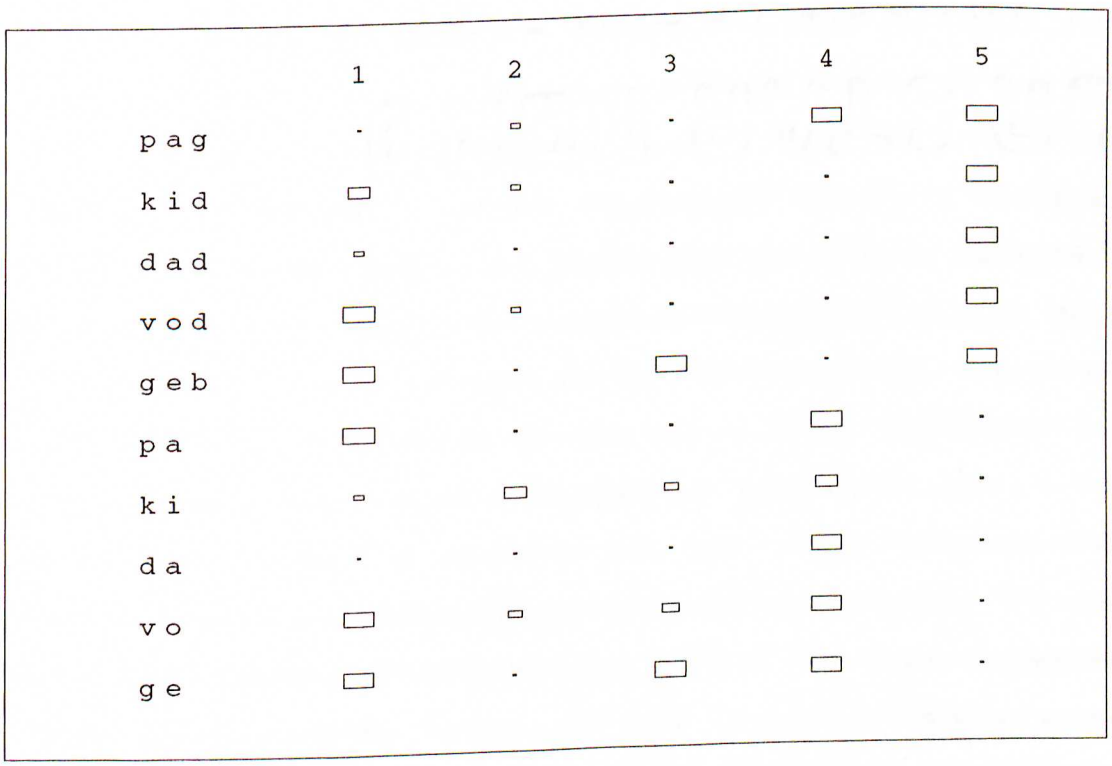

Figure 5.15: Box plot of hidden layer activations after the second segment was input during the production task in a "post-del" task run.

copying the value. But this is not the case. The unit turns on only after the second segment is presented for singular words.

5.9.4 Summary of the Results

Through analyses of the hidden layer representations during the English plural morpheme acquisition task, the non-assimilatory suffixation task, and the final deletion task, I have shown that the network indeed encodes a "plural" UR in a distributed representation ("distributed" in a sense of "superimposed" scheme). In the analyses of hidden representations during the first English-plural run and the non-assimilatory suffixation run, we were not able to find a **single** unit responsible for plurality; rather 2 units were involved, one for perception and the other for production. Also the encodings were found to be different for the three runs. In the first English-plural run the g-number unit was on for plural in the perception task, while the other unit was on for the singular nouns in the production task. In the second run the g-number unit was on for plural nouns during both perception and production tasks. The units responsible for g-number in the non-assimilatory suffixation run were on for singular nouns during both perception and production tasks. Despite the differences among these runs, I can safely say that there are certain units which are responsible for detecting the g-number. For a given morpheme, English plural or an artificial "plural", there is a commonality among the hidden layer patterns for all realizations of the morpheme. And in one case, the network chose to dedicate the same unit to representing plurality for both tasks (This may not have been a coincidence; the probability of choosing the same unit twice is around 0.15, considerably smaller than the observed value of 0.25 that I got). This is what is required for a UR to be part of "competence".

5.10 Test of Knowledge Transfer

Elaboration tolerance (McCarthy, 1988) is the ability of a model to be elaborated to take additional phenomena into account. One example of this phenomenon is the problem of *knowledge transfer*, how a pattern of activity learned by one network becomes interpretable by another. As Pinker and Prince (1988) point out, Rumelhart and McClelland's (1986) past tense model failed to account for the fact that homophonous verb roots may have different past tenses (e.g., *ring* and *wring*). Nor is the model "elaboration tolerant" enough to explain the commonality between the /t/-/d/-/ɪd/ alternations found in regular past tense forms and the /s/-/z/-/ɪz/ alternations found in the third person singular, regular plural nouns, possessive and so on. To partially answer this problem, some experiments were performed.

5.10.1 Method

Except for the connections to and from the meaning units, the network in these experiments is independent of morphemes. Input for the network at a given time step consists of a segment of a word and the meaning of the word. The segments vary, while the meaning stays same for the duration of a word's input. It is possible to train this network on the same rule twice with the same set of words, but with the different grammatical feature units. This is analogous to the "suffix" rule that can be applied to the third person singular verb as well as the plural noun. For example, not only is the plural of *top tops* as in "The spinning tops fell off the table," but the third person singular form is also *tops* as in "His batting average of 390 tops the previous record." It is also possible to train this network using the same set of words on different types of rules by, for example, adding different suffixes to the same stem. For example, the

progressive form of the verb *top* is *topping*. How much faster, then, can the network learn the second rule? Can it learn the similar rule faster than the different rule? If the answer is yes, it is evidence that this network is capable of transferring its prior knowledge to another environment that is similar. To test whether transfer would occur, I first trained a network on the suffix rule. I then added a new grammatical feature unit and trained the network either on the same rule for the new morpheme or a different (but equally difficult) rule. Of interest here was the relative time to learn the different rules. One hundred different data sets were generated randomly, each with 20 words. The numbers of epochs needed to learn the different rules for all 100 simulations were recorded, and the means were calculated and compared.

In the first experiment, two different rules were to be learned by a network. One rule was that of suffixing as described earler: an /s/ or /z/ is affixed to the uninflected basic form CVC, the suffix agreeing on the **voicing** feature with the previous segment (*Rule 1*). The other one was a made-up rule which added /z/ after /b/, /p/, /f/, or /v/, and /s/ after /t/, /d/, /k/, or /g/ (*Rule 2*). Rule 2 is harder than Rule 1, since in Rule 2 there is no commonality between the suffix and the final consonant of the stem where it is affixed, while in Rule 1 there exists a cue according to voicing.

The network possessed two units for the grammatical feature (say, one for the plurality and the other tense). The same set of stem words were used with the two different rules. In phase 1, only the first grammatical feature unit was involved in learning Rule 1. In phase 2, the other unit was learned: in one case, it was the same rule (Rule 1) as the phase 1 (analogous to applying the knowledge of the plural rule to the third person singular rule); in another case, the rule was different (Rule 2).

In the above described experiment, Rule 1 had an inherent advantage over Rule 2 in phase 2. Since in phase 1 the network saw only Rule 1, by default the network tended to think it was Rule 1 that it had to learn, causing Rule 2 to take more time

to learn than Rule 1. To give fair treatment to both rules, a second experiment was conducted in which there was no intrinsic rule associated with the stem words. In the second experiment, three different rules were employed: two as before with the addition of a new artificial rule which added /s/ after /p/, /b/, /t/, or /d/, and /z/ after /k/, /g/, /f/, or /v/ (*Rule 3*). Again Rule 3 is harder than Rule 1, while there are no differences in terms of difficulty between Rule 2 and Rule 3. The network had 3 separate grammatical feature units, one for each rule. In phase 1, the first two units were trained on Rule 1 and Rule 2. The training was done one rule at a time: at a given time, only one rule was learned on a unit. The network saw Rule 1 in the first half of an epoch, and Rule 2 in other half. This was to ensure that the network could not expect any particular rule to be associated with the stem words. The rule order was chosen randomly. In phase 2, all three different rules were learned on all three units. In this case new rule (Rule 3) was to be learned on the third unit.

5.10.2 Results

As seen in Figure 5.16, the first experiment clearly shows that it took less time to learn the similar task than the different task ($t = 2.350$, $df = 99$, $p < 0.021$). The results are not surprising, since the network was given enough help to make Rule 1 be learned easier then Rule 2 as explained above.

Figure 5.17 shows the results of the second experiment. Note that Rule 1 and Rule 2 were trained before, while Rule 3 was newly introduced. As expected, Rule 3 took the longest time to learn ($t = 3.240$, $df = 99$, $p < 0.0016$[11]). Rule 1 took a little more time than Rule 2 because the learning of the second rule took place later

[11]This was for the pair of "init2" and "different". No significant difference was found between "init1" and "init2"; $t = 1.123$, $df = 99$, $p < 0.228$.

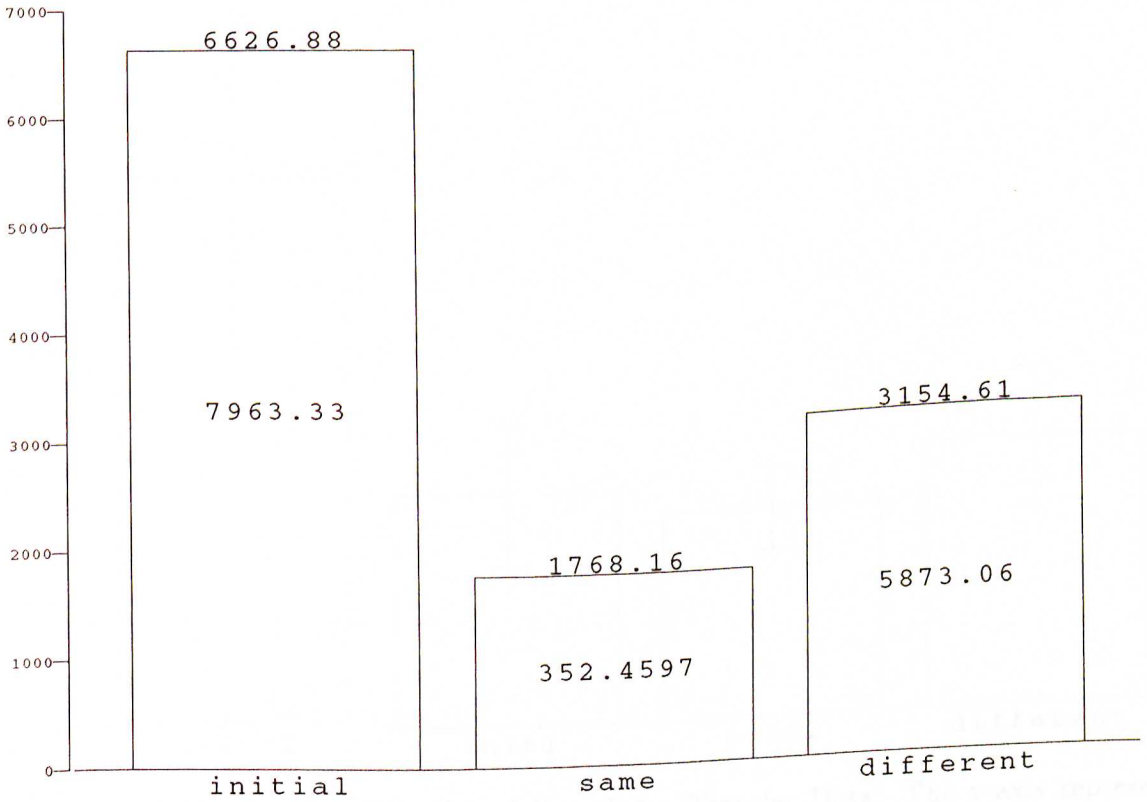

Figure 5.16: Bar plot of Knowledge Transfer Data. The abscissa is the time taken for the training, *initial* means the phase 1 training, *same* the phase 2 "same" kind of rule, *different* the phase 2 "different" kind of rule. The number above the bar denotes the average training time in word presentation over 100 simulations, and the number inside the bar is the standard deviation.

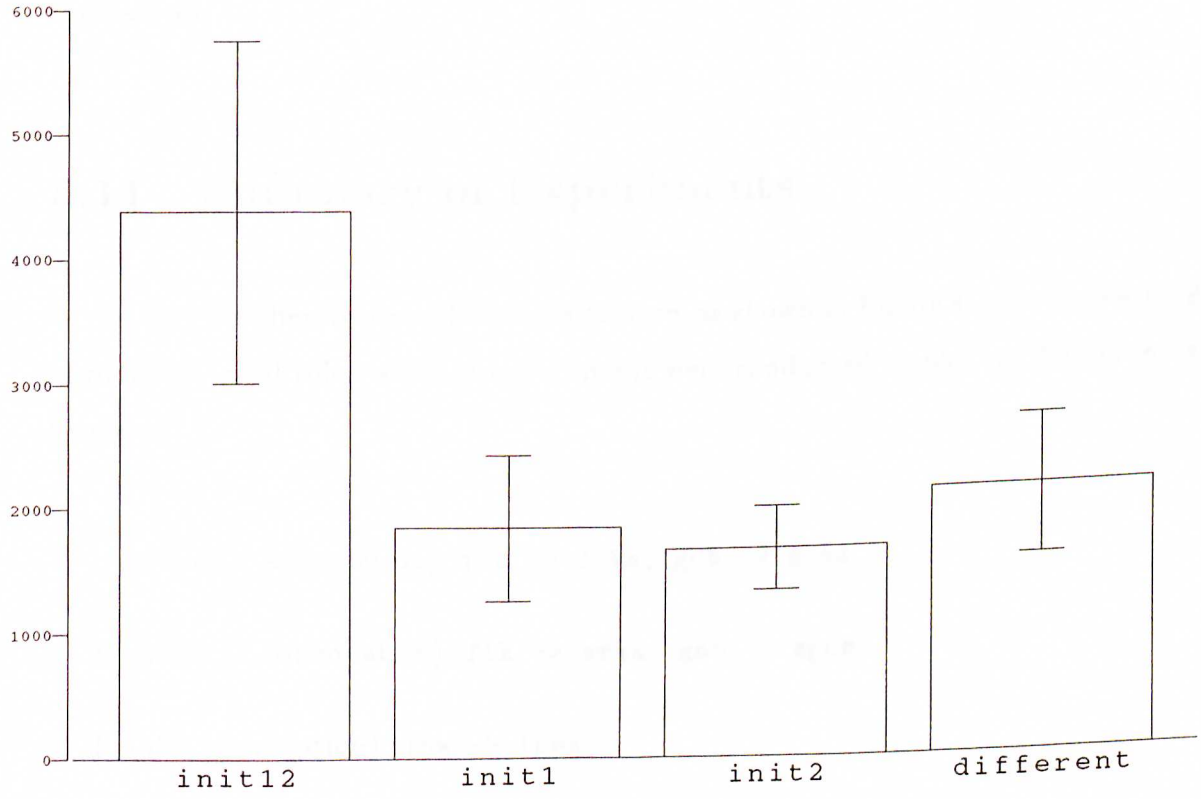

Figure 5.17: Another Bar plot of Knowledge Transfer Data. The y-axis represents the average time taken to learn the different rules. The length of the vertical line shows one positive and negative standard deviation away from the average. *Init12* represents the time required to learn phase 1 of Rule 1 on unit 1, and of Rule 2 on unit 2; *init1* represents the phase 2 training of the Rule 1 on unit 1; *init2* represents the phase 2 training of Rule 2 on unit 2; and *different* represents the phase 2 training of Rule 3 on unit 3.

and resulted in some unlearning. This was confirmed when a different order (Rule 2 in the first half and Rule 1 in the second half) was employed. This time Rule 2 took more time than Rule 1. It is the training order that made one rule take a little more time than the other when both of them were involved in phase 1, not the tasks themselves.

5.11 Summary of Experiments

To test the effectiveness of the architecture, as shown in Figure 4.1 for the learning of morphological rules, a set of experiments were conducted employing the following 9 rules:

1. suffix (+ assimilation): `fik -> fiks, gob -> gobz`
2. prefix (+ assimilation): `fik -> sfik, gob -> zgob`
3. infix (gemination): `ipa -> ippa`
4. initial deletion: `fik -> ik`
5. medial deletion: `ippa -> ipa`
6. final deletion: `fik -> fi`
7. tone change: `fík -> fìk`
8. reversal: `fik -> kif`
9. Pig Latin: `fik -> ikfe`

Table 5.10: Results of morphological process experiments.

	Production		Perception
	% Segments Correct	% Affixes Correct	% Plurality Correct
Suffix	82.3	82.5	79.0
Prefix	62.0	76.3	76.0
Infix	73.5	42.5	90.0
English Pl	93.9	87.5	100.0
Os Pl	80.0	75.0	75.5
Pre-del	12.5	-	60.0
Mid-del	23.8	-	73.8
Post-del	57.5	-	67.5
H-L Tones	97.5	-	99.1
Reversal	22.5	-	13.0
Pig Latin	27.2	-	61.3

The network succeeded on rule types which are common in human languages and failed on those which are rare or non-existent. Types 1 and 2 are common, types 3–7 less common, type 8 non-existent, and type 9 apparently encountered only in language games. The results are summarized in Table 5.10. There is a moderately high correlation ($r = 0.74$) between the results of the production task and those of the perception task. One notable exception is the results of the perception task for Pig Latin. The network performed quite well, even though this rule is found only in language games. Since each rule types was tested separately, the network could tell rather easily if the word being perceived was a stem word or a Pig Latin form just by attending to the final phoneme, vowel /e/.

When the network was tested on the English plural morpheme acquisition task, the network predicted the correct segments for all but 2 of the stem segments and 14 of the 16 suffixes in the two runs. When the plural morpheme appeared in the perception task, the output number unit behaved appropriately for all of the 16 test items.

The network successfully encoded UR "plurality" as a distributed representation on the hidden layer. For a given morpheme, such as plural, there is a commonality among the hidden layer patterns for all realizations of a morpheme. It has been shown that certain units on the hidden layer are responsible for encoding patterns that are analogous to URs.

The network was tested to see if it can exhibit elaboration tolerance. Two experiments were done to show that it could indeed show knowledge transfer. The network was able to learn a rule which was identical to the one it had already learned significantly more easily than one which was different.

6

Discussion

In the last chapter, detailed reports on the core of the thesis was given: what experiments were done, how the model was trained, and what results were obtained. In this chapter, I will examine whether all the research hypotheses were substantiated, explain some of the findings, and describe limitations of the study as well as directions for future extensions and research.

6.1 Tests of Research Hypotheses

The hypotheses enumerated in Chapter 2 proved to be correct. The model used in the current study accommodated both production and perception in a single network (Hypothesis 1). The model achieved this by having both form and meaning represented in the network, and performing the more psychologically plausible task of associating form with meaning. The experiments using various types of morphological processes showed that the model learned to exhibit the correct behavior relative to these processes without the benefit of pre-defined underlying representations or

explicit rules (Hypothesis 2). Rules are implicit in the connection weights between units which the network develops while trying to produce the correct outputs. Among the many rule types tested, the network succeeded on rule types which are common in human languages and failed on those which are rare or not encountered in human languages. Extensive analyses of the hidden representations suggest that the model encodes underlying representations as distributed representations on the hidden layer. Also, the model captured some central generalizations about sound patterns (Hypothesis 3). It was able to learn new rules which were identical to ones already learned much more easily than rules which were different from those already learned, showing its ability to transfer knowledge acquired from one task to another similar task.

6.2 Explaining the Language Universals

The model shows clear evidence of having learned morphological rules for both production and perception. The degree of mastery of the rules, at least for production, mirrors the extent to which the different types of rules occur in natural languages.

The study of language universals has been a major focus of modern linguistics for at least the past three decades. Why do languages share the universal properties that they do? Why do languages exhibit the range of variation that they do? Why are certain logically possible properties not found in any human languages? In attempting to answer these questions, it appears that linguists can be grouped according to four different theories, as explained by Hawkins (1989b):

> Some argue for the innateness of general linguistic principles housed within a language acquisition device (LAD) which enables the new-born child

> to acquire the particular language of his/her community with remarkable speed and despite impoverished input. Others argue for a more social, rather than a biological, foundation to language: the communicative (discourse-pragmatic) functions that language users perform are reflected in linguistic structure. Yet others appeal to the psychological demands placed upon language users in the production and comprehension of language in real time. These so-called 'processing' demands are also argued to be reflected in its structure, as are certain intrinsic properties of our human perceptual and cognitive apparatus. Finally, there are more grammar-internal explanations, whereby one part of the grammar is claimed to be explained by another, for reasons essentially of internal consistency. (p. 3)

My approach in this thesis is one that is based on the demands of learnability and processing (third approach from the above quote). I do not believe that the work described in this thesis necessarily makes strong claims that human perceptual processes are learned by the model (the model might not be the right one after all), but it gives an example of the kind of contribution connectionism can make to the search for language universals and their explanation. In the rest of this section, I will explain how the model has discovered some universals.

6.2.1 Affixation and Deletion

The network performed much better on the affixation tasks than on the deletion tasks. The reason might be that for the affixation tasks, the model had to predict the segment following the current phoneme, while for the deletion cases its task was to predict the phoneme that would come after the next one if it were not deleted. For the latter task, there is a gap between the current phoneme and the one that is to

be predicted, making it more difficult for the network to predict the correct segment. Consider the following three cases:

(6.1)	/tap/	+	/#/	and	/tap/	+	/s/
(6.2)	/tam/	+	/#/	and	/tam/	+	/z/
(6.3)	/tap/	+	/#/	and	/ta/	+	/#/

Problems (6.1) and (6.2) are somewhat easy, as in each case the identity of the last phoneme depends on the penultimate phoneme. The different contexts created by the penultimate phoneme are sufficient to ensure that different predictions can be made for the last phoneme. However, for (6.3), the final phoneme, that is, the word boundary, comes after the final phoneme in the stem for the singular case, while it comes after the second phoneme for the plural case. To predict the word boundary correctly in both cases, the network must develop different internal representations relative to the second phoneme for each case. (Indeed, the hidden unit activations have to be different if different outputs are to be produced.) Only in this way can the network generate the correct final phoneme (and in the next time step the word boundary) for the singular case and simply the word boundary for the plural case. Since the prediction tasks are the same for both cases up to the second phoneme, the network tends to develop the same hidden representations. This "homogenizing" process seems to strongly hinder learning in deletion tasks. Servan-Schreiber et al. (1988) report similar findings in their study on learning two arbitrary sequences of the same length and ending in two different letters such as:

PSSS P and TSSS T

They report:

> ...the predictions in each sequence are identical up to the last letter. As similar outputs are required on each time step, the weight adjustment procedure pushes the network into developing *identical* internal representations at each time step and for the two sequences – therefore going in the opposite direction than is required. (p. 29)

The very nature of the back-propagation learning rule and the structure of the model enable correct prediction of the affixed phonemes but make difficult the prediction of the segments after a phoneme is deleted. This is one explanation for the universal tendency of natural languages to exhibit many affixation processes, but few deletion processes.

6.2.2 Affixation and Assimilation

The network was able to generate the appropriate forms even in the prefix case when a "right-to-left" (anticipatory) rule was involved. That is, the fact that the network was trained only on prediction did not limit its performance to left-to-right (perseverative) rules since it had access to a static "meaning", permitting it to "look-ahead" to the relevant feature on the phoneme following the prefix. What makes this interesting is the fact that the meaning patterns bear no relation to the phonology of the stems. The connections between the stem meaning input units and the hidden layer units were being trained to encode the voicing feature even when, in the case of the test words, this was never required during training. For example, consider the

following training set of artificial data.

(6.4) FIK[1] + SINGULAR --> /fik/

(6.5) FIK + PLURAL --> /sfik/

(6.6) KOB + SINGULAR --> /kob/

When the network predicted the prefix of the word "KOB" for plural,

(6.7) KOB + PLURAL --> ?? + /kob/,

it had available to it the characteristics of the first phoneme in the stem: among them notably the voicing feature. The meaning "KOB" has /k/ associated with it as its first segment. Thus the network knows that it has to produce /s/, since the grammatical feature unit is on and /k/ is voiceless.

In any case, it is clear that right-to-left assimilation in a network such as this is more difficult to acquire than left-to-right assimilation, all else being equal. Cross-linguistic studies of morphology have revealed an asymmetry in the frequency of affixing processes in favor of suffixing over prefixing (Hawkins, 1988a), meaning that there are at least fewer opportunities for the right-to-left process.[2] I am unaware of any concrete evidence that would support left-to-right assimilation as easier than right-to-left assimilation, though in trying to explain the asymmetry between the processes Hawkins and Cutler (1988) argue that:

> ...the linguistic and psycholinguistic evidence together suggest that language structure reflects the preference of language users to process stems

[1] As explained before, the items in capitals represent meanings. Since the word is a made-up one, stem meaning is arbitrary.

[2] Of course, this does not necessarily mean that left-to-right rules are more common than right-to-left rules. In fact, right-to-left stress rules are more common than left-to-right ones.

> before affixes, in that the component preferred for prior processing receives the most salient (initial) position in the word, the component to be processes second a less salient position. That is, the suffixing preference results in stems generally being ordered before affixes because language users prefer to process stems before affixes. (p. 311)

Whatever reason there might be, it is very encouraging to see that the model performs in a way that mirrors human language: suffixation is more frequent across languages than prefixing, and both are considerably more frequent than infixing.

6.2.3 Reversal

What is it that makes the reversal rule, apparently difficult for human language learners, so difficult for the network? Some aspects of the rule were learned. In 49% of the cases the network produced a CVC syllable as the plural form. What it could not do was to predict the correct consonants for the past tense.

Consider what happens in the suffix or prefix cases for the production task. The input consists of the sequence of phonemes representing the stem of a word, together with the stem meaning seen during training and the plural, not seen in this combination during training. Given a novel set of patterns, the network treats it as a combination of two sorts of patterns it has seen before: one of which is a sequence of phonemes representing the stem of a word, excluding the affix, along with the stem meaning; the other of which is the plural input, along with the feature of the segment that determines the appropriate plural form. The relevant phonetic feature is readily available in the suffix case as a part of the input. In the prefix case, as argued in the previous subsection, it is available as an acquired property of the stem meaning.

For the reversal case, if we think of the novel item in the form of a *set* rather than a sequence, then exactly the same set of segments is used for both singular and plural words. More importantly, however, since the network's task is to predict the next segment, there can be no sharing at all between the singular and plural forms in terms of prediction. Patterns on the hidden layer develop in response to prediction, so we should expect little similarity between context inputs for singular and plural words. As a result, the network does not have much material available for interpreting the novel reversed words. Presented with the novel plural form, it is more likely to respond based on similarity with a word containing a similar sequence of phonemes (e.g., `gip` and `gif`) than respond with the correct mirror-image sequence.

It is important to note, however, that the network's difficulty with the reversal process does not necessarily presuppose the type of representations that result from training a simple recurrent network on prediction. Rather difficulty results more from the fact that the network is trained to map meaning to form and form to meaning, rather than form to form, as in the case of the RM model. Any network of the former type which represents linguistic form in such a way that the contexts of the phonemes are preserved is likely to exhibit this behavior.[3]

6.3 Limitations and Future Extensions

The model successfully learned morphological rules by making generalizations from the exemplars by changing the connection weights during the process of learning. The set of weights the model developed in producing the desired plural morphemes constrained the model's outputs to follow the desired patterns, and what looks like

[3] I am grateful to Dave Touretzky for helping to clarify this issue.

a rule-governed behavior is in fact embodied in these weights. However, the network does not yet achieve all that I would like.

6.3.1 Learning without Teacher Forcing

One of the most serious limitations of the current research is that the network was tested with the help of a teacher. That is, to test the network's performance on the production task, I gave the network the appropriate segments for the stem successively, along with the meaning of that stem and the g-number unit on for plural. I then examined the prediction output units at the point where the plural morpheme should appear. Thus the network was pushed along the right path through *teacher forcing*; The task was that of example (6.9) rather than (6.8) where no actual segments are given as input and the network should be able to produce the whole word, stem and suffix altogether.

(6.8) CHIP + PLURAL --> ??.

(6.9) CHIP + PLURAL --> /ʧɪp/ + ??.

To test performance on the perception task, I gave the network the sequence of input segments for a word, set the stem meaning units to the appropriate pattern, and set the g-number unit initially to 0.5, to indicate an unknown g-number. Upon the presentation of each new segment, the g-number unit was copied from the output on the previous time step. I then examined the output g-number unit after the appearance of either the appropriate plural form or the word boundary. Thus, the network's task was that of example (6.11) rather than (6.10) where no meaning is involved at all as a part of input and the network should perceive the correct meaning: s-meaning as well as g-number.

Table 6.1: Results of some suffixation experiments without teacher forcing.

Task	Group	Production	Perception
		% Segments	% Meaning Correct
/s/-/z/ Suffix	Training	99.6	98.1
	Testing	62.1	56.3
/s/ Suffix	Training	99.8	96.9
	Testing	69.6	76.8

(6.10) /ʃIps/ --> ??.

(6.11) /ʃIps/ --> CHIP + ??.

I have been able to train the network to generate entire "phonological" words given only their meanings and to produce complete stem meanings given only the phonological words, but this has not been possible with novel combinations, that is, for the plural forms of the test words. Results of some of the runs are summarized in Table 6.1.

As can be seen from the table, training data responded very well to both perception and production tasks in */s/-/z/ suffix* simulations (with the same experiment environment as described in Section 5.5 with the rule that adds /s/ after a voiceless consonant and /z/ after a voiced one), yet performance was poor on the test words. Note that learning without teacher forcing is very difficult, since the network has to remember the path to get the right segments up to the suffix and also must encode the suffixing rule to produce the right suffixes in the production task. In the perception task, the network must exhibit the ability to pick the correct meaning from many possible candidates, by weeding out the ones that do not have the segments in the word given so far, one step at a time until it can select one and only meaning denoting the given word.

To test if the network would perform better with easier rules, another experiment was done. The task was to add just an /s/ to a singular word to form a plural. Unlike previous tasks, the network does not need to encode variations of the suffix; it is sufficient for the network just to know that it has to affix an /s/ after the stem. As can be seen from the table, performance on the test words improved only slightly, even though the task was much easier. Further examination of data revealed that either the stem segments were correct (but the suffix was wrong), or the stem production was poor (but the suffix was correct). From this, I would assume that the network was able to encode only one piece of information, either the path information or the suffix rule. It might be the case that we are expecting too much from this simple architecture. The network has to learn two different kinds of things, the arbitrary meaning-stem mappings (rote) and the general rule for grammatical features. Can this network encode both pieces of information if it is given some help? In an effort to force the network to accommodate both kinds of information, I trained the network with a variable amount of noise added to the input data. Adding noise might help generalization, since it expands the training set. The new set has some points in it obtained by extrapolating from a given input to one near it (the "noisy" one). Since the original input and the noisy one are given the same output, successful training forces the network to reduce the degree to which it "learns" the data in the training set. So far the addition of noise has not made any improvements worth mentioning.

It is important to discover ways to make the system robust enough to respond appropriately to novel combinations of meanings and forms. In addition to the issues of addition of carefully designed noise which I tried without obtaining significantly better performance, another issue should be addressed: how much supervision is reasonable? In the experiments reported in this thesis, all of the simulations were done with the correct input data given from outside. The results were impressive.

Yet I could not get the network to perform well when asked to generate its own input. It is critical to determine how much external input is sufficient to carry out the task of producing a whole sequence of segments given only meaning and perceiving the meaning given only sequences of segments.

6.3.2 Rule Interaction

Equally as important as the network's ability to easily handle only those rules which occur in human languages, and the intended goal of learning without teacher learning, is the ability of the model to handle more complex phonological processes, as Touretzky and Wheeler's model does (see Touretzky and Wheeler, 1989, among others). An important question for future investigation concerns what happens in cases where the traditional analysis posits a sequence of rules operating on intermediate representations at different levels of abstraction. Ordered-rule interaction is the foundation of conventional generative phonology.

Is my network as it stands powerful enough to produce correct outputs in cases where rule interactions would occur in the generative phonological analysis? Do I need to make some modifications to the model to handle rule interactions? Would changing some of the parameters of the model affect the behavior of the network enough to accommodate these seemingly more difficult problems? Since I have not tried any more difficult problems involving rule interactions, I am not able to answer these questions. Modifying the training regimen so that learning would occur in stages might be enough: the network would learn the easier rules first and then be gradually introduced to the more difficult tasks. Elman (1991) shows that either when the input is presented incrementally, or when the network begins with a limited memory that gradually increases, the network is able to learn a complex grammar.

But my speculation at this time is that my model would have to have more hidden layers to correctly encode several rules and intermediate representations: the network might need to learn inherent rule orderings. For example, my network might be able to learn the rule interactions shown above with two hidden layers: one hidden layer which would encode the rules and underlying representations and another which would control the order of rule application. But precisely how this might work is still unclear.

Related to the rule interaction problem is the need for the model to explain more complex morphological phenomena than those encountered in the experiments described in this thesis. First, it should be able to account for *reduplication*, since the high frequency of reduplication in human languages implies that the model should be able to handle this kind of process. Reduplication is a morphological process that involves repetition of a part or all of a word to form a related word. We can easily find examples from English in children's speech: *moo-moo* (*cow*) and *woof-woof* (*dog*). In Bushman, a southern African language, the plural form of a noun is formed from the singular by reduplication, for example, /hɔro/ 'eggshell' becomes /hɔrohɔro/ 'eggshells' (Bergenholtz and Mugdan, 1979, p.60). For the model to be able to perceive and produce words using this process, the current model may need to be substantially modified. Reduplication requires a primacy-oriented STM: such a model would have to focus on the beginnings of sequences because without the first segment, it would be very difficult, if not impossible, to predict the rest of the sequence or to perceive the meaning of the word. Once a cue was input, the network would have to reproduce the part it had seen so far (production), or given a word, it would need to compare the different parts of morphemes to determine the meaning of the word (perception). A static representation of segments that encodes sequences, not just the contexts of the segments might be needed in this case. This

static representation, then, could be used to reproduce the desired sequence when given the correct cue or to determine the meaning when compared to the current sequence. The current model is designed to encode the contexts of segments, and its STM is therefore recency-oriented, since it is trained to predict the next segment; yet I am not sure if it would accommodate a static representation. The model might need some kind of help to acquire some knowledge of the syllable structure, as well as other features. Gasser (1991) reports an experiment in which a pair of SRNs was trained on a simple reduplication rule: one SRN was trained to predict the next segment, while the other's task was to predict the next syllable. He reports that the results were clearly far better than chance.

Another morphological process that is very challenging to the current model is that of *metathesis*, the transposition of two phonemes in a word. For example, in Sierra Miwok, a Penutian language of California, a basic form of the verb *celku*, 'to quit', is realized as [celukk] after undergoing metathesis (and gemination) (Goldsmith, 1990, pp. 83–87). As demonstrated by the Pig Latin experiment, this kind of process might be extremely difficult, if not impossible, for the model to acquire.

6.3.3 Input Corpus

The experiments reported here were carried out on only a small, and severely restricted input corpus. Only 20 CVC words were considered in each simulation run. To be able to claim the plausibility of this model as an adequate system that can process morphological phenomena, I need to expand my research to a bigger data set.

Unlike the RM model, my model did not treat both irregular and regular forms and can shed no new light on the question of whether networks can mimic the stages

that humans tend to go through in making linguistic generalizations, an important issue addressed by the RM model and in subsequent work by Plunkett and Marchman. Even though there is no reason why the model cannot account for this phenomenon, I have no way of knowing at this time what stages it would pass through.

Except for the English plural acquisition task, I used artificial language data for all the exeriments, partly because it was simpler. One of the biggest drawbacks of this approach is that it cannot take into account the phonotactics of real human languages.

6.3.4 Training Regimen

There were some unrealistic procedures employed in training. The network was given only noiseless input data. However, in reality, it is not possible to receive all words without any noise at all. The network was also given a priori word boundaries. A more realistic task would be to present the words in succession without any boundaries, so that the network would learn to detect the word boundaries and process them accordingly. If the model were given raw speech data, the two procedures of filtering and segmentation of some sort would need to play very important roles in perception. In fact, Kaye (1989) has argued that one of the primary functions of phonology is to facilitate parsing with segmentation:

> Clearly, human beings come equipped with a parser, and part of linguistics consists of gaining an understanding of how it might work. ... which requires analyzing a speech signal into linguistically relevant units such as sentences, phrases, words, morphemes, and ultimately sound units. Discovering how the speech signal can be so segmented is of interest to

> phonologists ... phonological processes serve to facilitate parsing. (p. 50)

As discussed earlier in this section, the network can handle noisy input; and it is speculated that noise can actually help in generalization. As for segmentation, I think my model would be able to perform this task. Elman (1990) suggests that this type of network can indeed find word boundaries. He presented his network with a stream of words without word boundaries, one segment at a time. The task for the network at each point in time was to predict the next letter. The pattern of errors over time revealed that at the onset of each new word, the probability of error was high; as more of the word was received, the probability of error declined, as the sequence became increasingly predictable.

The model described in this thesis employed both meaning and form in a network to perform perception and production tasks. The network was given both meaning and form from the beginning and trained on both production and perception tasks from the onset. But here arises a problem of when and how the meanings were presented. In my model the meanings were presented to the network from the beginning without any reservations. Yet, it is quite possible that a person can hear words without perceiving their definite meanings at all. It would be a conceivable training procedure to expose the network only to the sequence of segments and let it develop its own representations for each word in the corpus. It would also be possible to introduce meanings only after the completion of a certain number of training epochs. Even though this model is not a semantic model, it would be desirable to present the model with more psychologically plausible data, by introducing the meaning later in the training stages.

There is another issue at stake here. What is the relationship between perception and production? How does the perception of a word aid in its correct production? It

is reasonable to assume that when a speaker learns to utter a word, he first learns to perceive the word correctly, produces it, and he corrects the utterance by comparing it to what he hears. In the current study no attempts were made to investigate how perception and production tasks interact to help one another. Research in children's first word acquisition (e.g. Bates, 1976) shows that single word comprehension begins at about 9 months (Infants usually start by responding to their own names) while production of single-word utterances does not start before the infant is 12–13 months old. There are at least two ways a model might deal with the fact that production generally follows perception: (1) to explicitly sequence the tasks, by finding the correct point in the training process at which to introduce the production task, or (2) to present both tasks simultaneously and discover that production turns out to be the harder task. The latter would be a preferable course because the difference would then 'emerge'. In the current study option (1) was not tried. When the network was presented both tasks (option 2), performance on perception task was not significantly higher than that on production tasks during early learning. This is not surprising at all, since the training procedures I adopted in my experiments give unfair advantage to the production tasks. The training I am doing is more realistic for the perception tasks than for the production tasks, since the targets are given. In a real perception task the context (at least sometimes) produces a target. But in production tasks a target does not normally exist; the environment does not correct the speaker when he says something wrong, although a lack of comprehension by the listener might. This is not the case in my experiment. Targets are given and wrong outputs are corrected: learning is supervised. In order to study the interaction between perception and production, the current training regimen should be reworked to treat both tasks equally.

6.3.5 The Evolution of a Network

One of the severe disadvantages of the current model is that the network was hand designed, especially in that the size of the hidden layer had to be empirically determined. This is one of the biggest drawbacks inherent to most connectionist models. My distant goal is to design a self-evolving network: given only the problem, the model would come up with the optimal network. There are two main approaches to such a task. In one method the network starts with a minimum number of nodes and creates more nodes dynamically when the need arises (Ash, 1989; Fahlman and Lebiere, 1990; Hanson and Pratt, 1989); in other approach the network starts from a pool of available nodes and eliminates the nodes, as well as the weights, that do not contribute to the solution of the problem (Sanger, 1989; Mozer and Smolensky, 1989). At the moment, I am not sure what kind of algorithms I could add to the model so that it could cause the network to evolve, but I am certain that the model should be able to address this important question.

6.4 Summary

Experiments with different morphological processes showed that the model accommodates both perception and production in a single network (Hypothesis 1) and can acquire implicit "rules" without the benefit of pre-defined underlying representations (Hypothesis 2). Also, the model did exhibit knowledge transfer (Hypothesis 3).

The model shows clear evidence of having learned morphological rules and its degree of mastery of the rules mirrors the extent to which the different types of rules occur in natural languages. The network performed much better on the affixation tasks than on the deletion tasks. The network was able to generate appropriate

forms even in the prefix case where a "right-to-left" rule was involved. The fact that the network was trained only on prediction does not limit it to left-to-right rules because it has access to a "meaning" which permits the required "look-ahead" to the relevant feature on the phoneme following the prefix. It failed to learn a reversal rule which is unknown in human language.

One of the most serious limitations of the current research is that the network was tested with the help of a teacher. It is important to discover ways to make the system robust enough to respond appropriately to novel combinations of meanings and forms. Equally important is developing the model to handle more complex processes, such as rule interactions, reduplication, and metathesis. Experiments should also be expanded to larger data sets. Input words must be presented with noise and without boundary symbols. The relationship between form and meaning, as well as the relationship between perception and production should be investigated. My eventual goal is a self-evolving network: given only the problem, the model should provide the optimal network.

7

Conclusion

7.1 Overall Achievements

Even though my model has not yet produced a sequence of segments given only a meaning or been able to select a meaning given only sequence of segments (comprising a word) for novel data, it is capable of abstracting generalizations from exemplars, thus eliminating the need for pre-defined abstract underlying representations and explicit rules, a major part of generative phonology. In addition, my model failed on just those types of rules that are rarely found in human languages.

In my model, analogues of underlying representations are encoded as distributed representations on the hidden layer developed by the network. As I clearly showed in Chapter 5, this underlying representation is a phonological one and is not simply an encoding of the input meaning feature PLURAL in the hidden layer.

This model answers many of the questions raised by Pinker and Prince (1988) regarding the RM model by employing a different type of architecture and input

representation (*cf.* Related Work section in Chapter 2).

First, the kind of *dynamic* representation employed in my model, together with more realistic tasks of production and perception, results in constraints on what processes can be learned. In particular, the system has difficulty in learning reversal processes, which are not found in any human languages and which the RM model was capable of learning.

Second, the model exhibits knowledge transfer. Following training for several cases, the network was able to learn a new rule which was identical to one it had already learned much more easily than one which was different from any it had already learned.

Third, since this model makes use of the association of form with meaning, it can easily distinguish phonologically ambiguous words such as *son* and *sun*, unlike other phonological models which do not ever include meaning in the input.

As far as I know, the study reported here is one of the first attempts to train a single recurrent network on both production and perception tasks. Both Dorffner (1990) and Chauvin (1988) trained non-recurrent networks to associate meanings with words, but these models failed to deal with the temporal properties of words.

I believe my study has shed some light on how a particular type of cognitive phenomenon can be accounted for without the use of explicit symbols or rules. It remains to be seen how much my approach to the relatively trivial processes dealt with must be modified to deal with more complex processes and the elaborate mechanisms for handling them posited by traditional phonologists and their connectionist descendants (e.g., Touretzky and Wheeler, 1990a).

7.2 Further Directions

Among linguists of different schools one thing that is generally agreed upon is that words are not unanalyzable atoms. They are composed of smaller units and there is a hierarchy from more abstract units to less abstract units as shown earlier in Figure 2.1. In the current study, my main concern was with the areas of phonology and morphology. I presented feature matrices corresponding to single phonemes as input to the network and showed that the network could learn morphological processes, thereby perceiving and producing the correct words given the appropriate form or meaning. As shown in the analysis of hidden representations analogous to underlying representations, it is safe to assume that the activations which follow the presentation of a feature matrix segment compromise a distributed representation of that phoneme. Following a cluster analysis demonstrating the network's natural categorization of phonemes (Elman and Zipser, 1988, describes how their networks developed internal representations corresponding to phonetic features), we might be able to use these internal representations as input for another higher level network, thus completing the chain shown in Figure 2.1. Similarly we might be able to construct representations for syllables, an issue which I bypassed in the current study. Even though the network is told nothing about where syllable boundaries occur, how many syllables there are in a given word, or even that syllables exist, it might be able to encode and use the syllable structures. Corina (1991) shows that his SRN can find syllable boundaries when given real speech data in the form of phonetic segments with no explicit word boundaries. My network might catch critical information about syllables by finding regularities when it is repeatedly exposed to the input data. While it is very difficult for the network to learn sequences of segments (rote learning), as demonstrated in Chapter 5, it can detect regularities in the input data rather easily. For example, even though the model failed to learn the reversal rule, in half of the cases it produced the

correct CVC syllable type as the plural form. As suggested by Elman (1990) regarding experiments on discovering the notion of 'word', the pattern of errors might provide a clue to the network as to what are the recurring patterns in the input corpus. Focusing attention on its errors during training might allow the network to make some decisions about the location of syllable boundaries. It is not difficult to imagine a language learner making predictions about what will come next; the success or failure of those predictions could be used to identify the basic elements of the language being learned, including syllables. The force at work in this case might be a quantitative mechanism rather than a qualitative one, depending in part on the statistics of co-occurrence.

How then can we use the analyzed hidden representations or syllable structures discovered by the network? Do we need another higher level network? If a hierarchy of SRNs is needed, then how can we decide upon the levels? How many different sequence types should there be? And how many sequences of a given type should there be? Gasser (1991) proposes hierarchies of SRNs for segments, syllables, metrical feet, and phonological phrases. He posits that the hierarchy can be created with the help of some form of unsupervised learning, yielding sequence summary representations that satisfy perception and production constraints on each level.

Jacobs and Nowlan (Jacobs et al., 1990; Jacobs, 1990; Jacobs et al., 1991; Nowlan, 1990) and Śmieja (Śmieja, 1991) approach this kind of problem with a modular connectionist architecture. In Jacobs and Nowlan's approach, the networks composing the architecture compete to learn the training patterns. As an outcome of competition, different networks learn different training patterns and, thus, learn to compute different functions. The architecture performs task decomposition in the sense that it learns to partition a task into two or more functionally independent tasks and assigns distinct networks to learn each task. In addition, the architecture tends to allocate

to each task the network whose topology is most appropriate to that task. Śmieja describes a similar architecture in which a a type of "neural expert" employs a collection of modules which have the same processing capabilities, but which vary with respect to the particular aspects of the problem domain for which they are specialized.

What I need is an architecture that has the capability to add more units and to organize them into specialized subnetworks when new language functions are encountered. It might start out as a network able to process low end input, and when certain of having enough evidence for a new level, the central controller would make some adjustments to the network and add new units, thus constructing a new subnetwork. This new network would receive its input from the previous network in the form of internal representations. The networks would update their weights asynchronously. When this subnetwork had accumulated enough proof of the existence of a new level, then the central controller would step in to build another new subnetwork. This procedure would continue until the model was able to recognize a whole utterance. The accumulation of evidence could be accomplished by monitoring error patterns in the hidden layer. This is a highly speculative notion and needs a lot of polishing before it can be tested. What seems to be true is that the speech signal is endowed with an intrinsic regularity at several levels and there is a need to have a model that can accommodate this hierarchy of regularity.

A related problem is that, as shown in Chapter 5, I presented words in isolation, as presegmented signals labeled with phonetic features, which is not the case for real speech data. If a hierarchy of SRNs is to emerge naturally, then input should be an uninterrupted stream of raw speech data. It should filter out noise from the speech signal, and segment the signal into different categories (phonemes) according to appropriate (phonetic) features. It should discover for itself the existence of the basic units of language, including the syllable, the morpheme, and the word. It

should analyze distributed representations on the hidden layer, use them to feed the next level in the hierarchy, and add new levels as need arises. Once it has finished being trained in perceiving the input signal, the resulting network should then be able to perform production tasks. The quest for these answers will be my ultimate goal in the process of my research. The study presented in this thesis is a right step in this direction and hopefully will be a significant component of a complete speech understanding system.

References

Allen, R. (1989). Adaptive training of connectionist state machines. In *Proceedings of ACM Computer Science Conference* (p. 428).

Allen, R. (1990). Connectionist language users. *Connection Science, 2,* 279–311.

Ash, T. (1989). *Dynamic node creation in backpropagation networks.* (Technical Report ICS Report 8901). Institute for Cognitive Science, University of California, San Diego.

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

Bates, E. (1976). *Language and Context.* New York: Academic Press.

Bear, J. (1988). Morphology with two-level rules and negative rule features. In *Proceedings of COLING '88* (pp. 28–31).

Bergenholtz H. and Mugdan J. (1979). Einführung in die Morphologie. Stuttgart: Verlag W. Kohlhammer.

Camargo, F. (1990). *Learning algorithms in neural networks.* (Technical Report CUCS-062-90). Computer Science Department, Columbia University, New York, NY.

Carson, J. (1988). Unification and transduction in computational phonology. In *Proceedings of COLING '88* (pp. 106–111).

Chalmers, D. (1990). Why Fodor and Pylyshyn were wrong: The simplest refutation. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 340–347). Hillsdale: Lawrence Erlbaum Associates.

Chauvin, Y. (1988). *Symbol acquisition in humans and neural (PDP) networks.* PhD dissertation, University of California, San Diego.

Chauvin, Y. (1989). Toward a connectionist model of symbolic emergence. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 580–587). Hillsdale: Lawrence Erlbaum Associates.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Cambridge: MIT Press.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English.* New York: Harper and Row.

Church, K. (1987). Phonological parsing and lexical retrieval. *Cognition, 25,* 53–69.

Cleeremans, A., Servan-Schreiber, D., and McClelland, J. (1989). Finite state automata and simple recurrent networks. *Neural Computation, 1,* 372–381.

Corina, D. (1991). *Towards an understanding of syllable structure: evidence from linguistic, psycholinguistic and computational investigations of the syllable.* PhD dissertation, University of California, San Diego.

Cottrell, G. and Tsung, F.-S. (1989). Learning simple arithmetic procedures. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 58–65). Hillsdale: Lawrence Erlbaum Associates.

Cullingford, R. (1981). SAM. In R. Schank and C. Riesbeck (Eds.), *Inside Computer Understanding.* Hillsdale: Lawrence Erlbaum Associates.

Davis, S. (1991). Language games. In *The Encyclopedia of Language and Linguistics.* New York: Pergammon Press.

Dinnsen, D. (1979). Atomic phonology. In D. Dinnsen (Ed.), *Current Approaches to Phonological Theory* (pp. 31–49). Bloomington: Indiana University Press.

Dorffner, G. (1990). *A sub-symbolic connectionist model of basic language functions.* PhD dissertation, Indiana University, Bloomington, IN.

Donegan, P. and Stampe, S. (1979). The study of natural phonology. In D. Dinnsen (Ed.), *Current Approaches to Phonological Theory* (pp. 126–173). Bloomington: Indiana University Press.

Elman, J. (1989a). *Representation and structure in connectionist models.* (Technical Report 8903). Center for Research in Language, University of California, San Diego.

Elman, J. (1989b). Structured representations and connectionist models. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 17–25). Hillsdale: Lawrence Erlbaum Associates.

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

Elman, J. (1991). *Incremental learning, or the importance of starting small.* (Technical Report 9101). Center for Research in Language, University of California, San Diego.

Elman, J. and Zipser, D. (1988). Learning the hidden structure of speech. *Journal of the Acoustical Society of America, 83,* 1615–26.

Elson B. and Pickett V. (1962). An Introduction to Morphology and Syntax. Santa Ana: Summer Institute of Linguistics.

Fahlman, S. (1988). Faster-learning variations on back-propagation: An empirical study. In *Proceedings of the 1988 Connectionist Summer School* (pp. 38–51). San Mateo: Morgan Kauffmann

Fahlman, S. and Lebiere, C. (1990). The cascade-correlation learning architecture. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, San Mateo: Morgan Kaufmann.

Feldman, J. and Ballard, D. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205–254.

Fodor, J. (1976). *The Language of Thought.* Sussex: Harvester Press.

Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28*, 3–71.

Fromm, H. (1982). *Finnische Grammatik.* Heidelberg: Carl Winter Universitätsverlag.

Gasser, M. (1991). Hierarchies of simple recurrent networks for word recognition and production . Paper presented at AAAI Spring Symposium on Connectionist Natural Language Processing.

Gasser, M. and Lee, C.-D. (1990). Networks that learn about phonological feature persistence. *Connection Science, 2*, 265–278.

Gasser, M. and Lee, C.-D. (1991). A short-term memory architecture for the learning of morphophonemic rules. In R. Lippmann, J. Moody, and D. Touretzky (Eds.), *Advances in Neural Information Processing Systems 3* (pp. 605–611). San Mateo: Morgan Kaufmann.

Gazdar, G. (1985). Review article: finite state morphology. *Linguistics, 23*, 597–607.

Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in Lisp: An Introduction to Computational Linguistics.* Reading: Addison-Wesley.

Goldsmith, J. (1989). Autosegmental licensing, inalterability, and harmonic rule application. In R. Graczyk, B. Music, and C. Wiltshire (Eds.), *Papers from the 25th Annual Regional Meeting of the Chicago Linguistic Society.*

Goldsmith, J. (1990). *Autosegmental and Metrical Phonology.* Cambridge: Basil Blackwell.

Goldsmith, J. (1991). Phonology as an intelligent system. In D. Napoli, and J. Kegl (Eds.), *Bridges between Psychology and Linguistics: a Swarthmore Festschrift for Lila Gleitman.* Hillsdale: Lawrence Erlbaum Associates.

Goldsmith, J. (to appear). Local modeling in phonology. In S. Davis (Ed.), *Connectionism: Theory and Practice.* Vancouver: University of British Columbia Press.

Goldsmith, J. and Larson, G. (1990). Local modeling and syllabification. In K. Deaton, M. Noske, and M. Ziolkowski (Eds.), *Papers from the 26th Annual Regional Meeting of the Chicago Linguistics Society.*

Hanson, S. and Pratt, L. (1989). *Comparing biases for minimal network construction with back-propagation.* (Technical Report CSL Report 36). Cognitive Science Laboratory, Princeton University, Cambridge, MA.

Hare, M. (1990a). Constraints on assimilation in vowel harmony languages. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 364–371). Hillsdale: Lawrence Erlbaum Associates.

Hare, M. (1990b). The role of similarity in Hungarian vowel harmony: a connectionist account. *Connection Science, 2,* 123–150.

Hare, M., Corina, D., and Cottrell, G. (1990). A connectionist perspective on prosodic structure. In *Center for Research in Language Newsletter,* 4–16.

Harnad, S. (1990). The symbol grounding problem. *Physica D, 42,* 335-346.

Harris, C. and Elman, J. (1989). Representing variable information with simple recurrent networks. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 635–642). Hillsdale: Lawrence Erlbaum Associates.

Hawkins, J. (Ed.) (1988a). *Explaining Language Universals.* Oxford: Basil Blackwell.

Hawkins, J. (1988b). Explaining Language Universals. In J. Hawkins (Ed.), *Explaining Language Universals* (pp. 3–28). Oxford: Basil Blackwell.

Hawkins, J. and Cutler, A. (1988). Psychological factors in morphological asymmetry. In J. Hawkins (Ed.), *Explaining Language Universals* (pp. 280–317). Oxford: Basil Blackwell.

Hertz, J., Krogh, A., and Palmer R. (1991). *Introduction to the Theory of Neural Computation.* Redwood City: Addison-Wesley.

Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence, 40,* 185–234.

Jacobs, R. (1990). *Task decomposition through competition in a modular connectionist architecture.* (Technical Report COINS TR 90-44). Department of Computer and Information Science, University of Massachusetts, Amherst.

Jacobs, R., Jordan, M., and Barto, A. (1990). *Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks.* (Technical Report COINS TR 90-27). Department of Computer and Information Science, University of Massachusetts, Amherst.

Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation, 3,* 88-97.

Jordan, M. (1986a). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 531–546). Hillsdale: Lawrence Erlbaum Associates.

Jordan, M. (1986b). *Serial order.* (Technical Report 8604). Institute for Cognitive Science, University of California, San Diego.

Karttunen, L. (1983). KIMMO: a general morphological processor. *Texas Linguistic Forum, 22*, 165–186.

Karttunen, L. and Wittenburg, K. (1983). A two-level morphological analysis of English. *Texas Linguistic Forum, 22*, 217–228.

Kataja, L. and Koskenniemi, K. (1988). Finite-state description of Semitic morphology: A case study of ancient Akkadian. In *Proceedings of COLING '88* (pp. 313–315).

Kaye, J. (1989). *Phonology: a Cognitive View.* Hillsdale: Lawrence Erlbaum Associates.

Kenstowicz, M. and Kisseberth, C. (1979). *Generative Phonology.* New York: Academic Press.

Kolen, J. and Pollack, J. (1990a). Back propagation is sensitive to initial conditions. *Complex Systems, 4*, 269–280.

Kolen, J. and Pollack, J. (1990b). Scenes from exclusive-or: Backpropagation is sensitive to initial conditions. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 868–875). Hillsdale: Lawrence Erlbaum Associates.

Koskenniemi, K. (1983a). Two-level model for morphological analysis. In *IJCAI-83* (pp. 683–685).

Koskenniemi, K. (1983b). *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production.* (Publication No. 11). Department of General Linguistics, University of Helsinki, Helsinki, Finland.

Koskenniemi, K. (1984). A general computational model for word-form recognition and production. In *COLING-84* (pp. 178–181).

Koskenniemi, K. and Church, K. (1988). Complexity, two-level morphology and Finnish. In *Proceedings of COLING '88* (pp. 335–340)

Kushner, M., Cleeremans, A., and Reber, A. (1991). Implicit detection of event interdependencies and a PDP model of the process. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society.* Hillsdale: Lawrence Erlbaum Associates.

Kwasny, S. and Faisal, K. (1990). Connectionism and determinism in a syntactic parser. *Connection Science, 2,* 63–82.

Lachter, J. and Bever, T. (1988). The relationship between linguistic structure and associative theories of language learning: A constructive critique of some connectionist learning models. *Cognition, 28,* 195–247.

Lakoff, G. (1988a). A suggestion for a linguistics with connectionist foundations. In *Proceedings of the 1988 Connectionist Models Summer School* (pp. 301–314). San Mateo: Morgan Kaufmann.

Lakoff, G. (1988b). Cognitive phonology. Paper presented at the Annual Meeting of the Linguistics Society of America.

Lang, K., Waibel, A., and Hinton, G. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks, 3,* 23–43.

Langacker, R. (1987). *Foundations of Cognitive Grammar, Volume 1.* Stanford: Stanford University Press.

Le Cun, Y. (1985). Une procedure d'apprentissage pour reseau a sequil assymetique. In *Proceedings of Cognitiva '85* (pp. 599–604).

Lee, C.-D. and Gasser, M. (1990). Learning morphophonemic processes without explicit rules and underlying representations. In *Proceedings of Seoul International Conference on Natural Language Processing (SICONLP '90)* (pp. 332–338).

Lee, C.-D. and Gasser, M. (1992). Where do underlying representations come from?: A connectionist approach to the acquisition of phonological rules. To appear in J. Dinsmore (Ed.), *Closing the Gap: Symbolicism vs. Connectionism.* Hillsdale: Lawrence Erlbaum Associates.

Marchman, V. and Plunkett, K. (1989). Token frequency and phonological predictability in a pattern association network: Implications for child language acquisition. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 179–187).

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language.* Cambridge: MIT Press.

Marcus, G., Ullman, M., Pinker, S., Hollander, M., Rosen, J., and Xu, F. (1990). *Overregularization.* (Technical Report Occasional Paper #41). Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

McCarthy, J. (1988). Epistemological challenges for connectionism. *The Behavioral and Brain Sciences, 11,* 44. Commentary to Smolensky (1988).

McClelland, J. and Rumelhart, E. (Eds.) (1986). *Parallel Distributed Processing,* Volume 2. Cambridge: MIT Press.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The Psychology of Computer Vision.* New York: McGraw-Hill.

Mozer, M. (1990). Discovering faithful 'Wickelfeature' representation in a connectionist network. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 356–363). Hillsdale: Lawrence Erlbaum Associates.

Mozer, M. and Smolensky, P. (1989). *Skeletonization: A technique for trimming the fat from a network via relevance assessment.* (Technical Report CU-CS-421-89). Department of Computer Science, University of Colorado, Boulder.

Newell, A. and Simon, H. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM, 19,* 113–126.

Nowlan, S. (1988). Gain variation in recurrent error propagation networks. *Complex Systems, 2,* 305–320.

Nowlan, S. (1990). *Competing experts: an experimental investigation of associative mixture models.* (Technical Report CRG-TR-90-5). Department of Computer Science, University of Toronto, Toronto, Canada.

O'Grady, W., Dobrovolsky, M., and Aronoff, M. (Eds.) (1989). *Contemporary Linguistics.* New York: St. Martin's Press.

Parker, D. (1985). *Learning-logic.* (Technical Report TR-47). Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA.

Pearlmutter, B. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation, 1,* 263–269.

Pearlmutter, B. (1990). *Dynamic recurrent neural networks.* (Technical Report CMU-CS-90-196). School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Pinker, S. and Mehler, J. (Eds.) (1988). *Connections and symbols.* Cambridge: MIT Press.

Pinker, S. and Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition, 28,* 73–193.

Plunkett, K. and Marchman, V. (1989). *Pattern association in a back propagation network: Implications for child language acquisition.* (Technical Report 8902). Center for Research in Language, University of California, San Diego.

Plunkett, K. and Marchman, V. (1990). *From rote learning to system building.* (Technical Report 9020). Center for Research in Language, University of California, San Diego.

Pollack, J. (1990). Electronic discussion on "Connectionists mailing list", December 20, 1990.

Port, R. (1990). Representation and recognition of temporal patterns. *Connection Science, 2,* 151–176.

Quillian, R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic Information Processing.* Cambridge: MIT Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65,* 386–408.

Rosenblatt, F. (1959). Two theorems of statistical separability in the perceptron. In *Mechanisation of thought processes: Proceedings of a symposium held at the National Physical Laboratory, November 1958, Vol. 1* (pp. 421–456). London: HM Stationery Office.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* New York: Spartan.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart and J. McClelland (Eds.), *Parallel Distributed Processing,* Volume 1 (pp. 319–362). Cambridge: MIT Press.

Rumelhart, D. and McClelland, J. (1986a). *Parallel Distributed Processing,* Volume 1. Cambridge: MIT Press.

Rumelhart, D. and McClelland, J. (1986b). On learning the past tense of English verbs. In J. McClelland and D. Rumelhart (Eds.), *Parallel Distributed Processing,* Volume 2 (pp. 216–271). Cambridge: MIT Press.

Sanger, D. (1989). Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. *Connection Science, 1,* 115–138.

Schank, R. (1975). *Conceptual Information Processing.* Amsterdam: North-Holland.

Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding.* Hillsdale: Lawrence Erlbaum Associates.

Sejnowski, T. and Rosenberg, C. (1986). *Nettalk: A parallel network that learns to read aloud.* (Technical Report JHU/EECS-86/01). Department of Electrical Engineering and Computer Science, The Johns Hopkins University.

Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems, 1,* 145–168.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. (1988). *Encoding sequential structure in simple recurrent networks.* (Technical Report CMU-CS-88-183). School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. (1989). Learning sequential structure in simple recurrent networks. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 1* (pp. 643–652). San Mateo: Morgan Kaufmann.

Small, S. (1990). Learning lexical knowledge in context: experiments with recurrent feed forward networks. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 479–486). Hillsdale: Lawrence Erlbaum Associates.

Śmieja, F. (1991). Multiple network systems (Minos) modules: Task division and module discrimination. In *Proceedings of the 8th AISB conference on Artificial Intelligence.*

Smolensky, P. (1988). On the proper treatment of connectionism. *The Behavioral and Brain Sciences, 11*, 1–74.

Sommer, B. (1980). The shape of Kunjen syllables. In D. Goyvaerts (Ed.), *Phonology in the 80's.* Ghent: Story-Scientia.

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

Tank, D. and Hopfield, J. (1987). Neural computation by concentrating information in time. In *Proceedings of the National Academy of Sciences, 84*, 1896–1900.

Touretzky, D. (1989). Toward a connectionist phonology: The "Many Maps" approach to sequence manipulation. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 188–195). Hillsdale: Lawrence Erlbaum Associates.

Touretzky, D., Elvgren III, G., and Wheeler, D. (1990). Phonological rule induction: an architectural solution. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 348–355). Hillsdale: Lawrence Erlbaum Associates.

Touretzky, D. and Wheeler, D. (1990a). A computational basis for phonology. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, San Mateo: Morgan Kaufmann.

Touretzky, D. and Wheeler, D. (1990b). Rational for a 'Many Maps' phonology machine. In *Proceedings of the Tenth European Meeting on Cybernetics Systems Research*. World Scientific Publishing Co.

Touretzky, D. and Wheeler, D. (1990c). Two derivations suffice: the role of syllabification in cognitive phonology. In C. Tenny (Ed.), *The MIT parsing volume, 1989-1990*. MIT Center for Cognitive Science, Parsing Project Working Papers 3.

van Gelder, T. (1989). Compositionality and the explanation of cognitive processes. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 34–41). Hillsdale: Lawrence Erlbaum Associates.

van Gelder, T. (1990). Compositionality: A connectionist variation on a classical theme. *Cognitive Science, 14*, 355–384.

van Gelder, T. (1991). Distribution and its advantages. Electronic discussion on "Connectionists mailing list", June 17, 1991.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1988). Phoneme recognition: Neural networks vs. hidden markov models. In *Proceedings of the ICASSP* (pp. 107–110).

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD dissertation, Harvard University, Cambridge, MA.

Wheeler, D. and Touretzky, D. (1989). *A connectionist implementation of cognitive phonology.* (Technical Report CMU-CS-89-144). School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Wheeler, D. and Touretzky, D. (1990). From syllables to stress: a cognitively plausible model. In K. Deaton, M. Noske, and M. Ziolkowski (Eds.), *Papers from the 26th annual regional meeting of the Chicago Linguistics Society, Part two: parasession on the syllable in phonetics and phonology.*

Wickelgren, W. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review, 76,* 1–15.

Widrow, G. and Hoff, M. (1960). Adaptive switching circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4* (pp. 96–104).

Williams, R. and Zipser, D. (1988). *A learning algorithm for continually running fully recurrent neural networks.* (Technical Report 8805). Institute for Cognitive Science, University of California, San Diego.

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

Williams, R. and Zipser, D. (1990). *Gradient-based learning algorithms for recurrent connectionist networks.* (Technical Report NU-CCS-90-9). College of Computer Science, Northeastern University, Boston, MA.

Woods, W. (1970). Transition network grammars for Natural Language Analysis. *Communications of the ACM, 13,* 591–606.

A

Chomsky-Halle feature matrix used in the study

Each segment in the data consisted of a binary vector representing modified Chomsky-Halle phonetic features. (Chomsky and Halle, 1968): 1 for the presence of a particular feature and 0 for its absence. Each segment type is uniquely specified by its vector. The distinctive feature matrix includes the following:

1. In Chomsky and Halle (1968), high, low, and back are defined with respect to a reference point that approximates the placement of the tongue for the vowel of the word *bed*.
   (a) **High** vowels are articulated by raising the tongue body above this point.
   (b) **Low** vowels by lowering the tongue body below this point.
   (c) **Back** vowels by retraction from this point.
2. **Anterior** sounds are produced with a constriction in front of the alveopalatal region.
3. **Coronal** sounds involve raising the blade of the tongue above its neutral position.
4. **Continuant** sounds permit air to flow through their stricture.

5. **Voiced** sounds are produced with vocal cord vibration.

6. **Nasal** sounds are produced with the velum lowered, thus allowing air to flow through the nose.

7. **Strident** sounds are marked acoustically by greater noise.

8. **Rounded** sounds are produced with a narrowing of the opening of the lips.

9. **Tense** sounds are produced with a gesture involving considerable muscular effort, or tension.

10. **Consonantal** sounds are produced with a radical obstruction in the midline of the vocal tract.

11. **Vocalic** sounds are produced with an *oral* cavity in which the most radical constriction does not exceed that found in the high vowels [i] and [u] and with vocal cords that are positioned so as to allow spontaneous voicing.

Each segment type is uniquely specified as a binary vector of 8 or of 13 depending on the tasks as shown in Figure A.1 and Figure A.2.

```
                features 8

vocalic high back anterior coronal voice strident round

                classification 16

p       0 0 0 1 0 0 0 0   ; pepper lip
b       0 0 0 1 0 1 0 0   ; baby rib
t       0 0 0 1 1 0 0 0   ; tie attack
d       0 0 0 1 1 1 0 0   ; did adder
k       0 1 1 0 0 0 0 0   ; cook ache
g       0 1 1 0 0 1 0 0   ; go big
f       0 0 0 1 0 0 1 0   ; fifty cuff
v       0 0 0 1 0 1 1 0   ; vivid give
s       0 0 0 1 1 0 1 0   ; source less
z       0 0 0 1 1 1 1 0   ; zone raise
i       1 1 0 0 0 0 0 0   ; each free
e       1 0 0 0 0 0 0 0   ; made
a       1 0 1 0 0 0 0 0   ; father cart
u       1 1 1 0 0 0 0 1   ; ooze too
o       1 0 1 0 0 0 0 1   ; bone know
#       0 0 0 0 0 0 0 0   ; word-boundary-marker
```

Figure A.1: Phonemes represented as binary vectors according to the Chomsky-Halle feature matrix.

```
                    features 13
vocalic consonantal high back low anterior coronal round tense
voice continuant nasal strident
                    classification 37
p  0 1 0 0 0 1 0 0 0 0 0 0 0  ; pepper lip
b  0 1 0 0 0 1 0 0 0 1 0 0 0  ; baby rib
t  0 1 0 0 0 1 1 0 0 0 0 0 0  ; tie attack
d  0 1 0 0 0 1 1 0 0 1 0 0 0  ; did adder
C  0 1 0 0 0 0 1 0 0 0 0 0 1  ; chin church
j  0 1 0 0 0 0 1 0 0 1 0 0 1  ; job judge
k  0 1 1 1 0 0 0 0 0 0 0 0 0  ; cook ache
g  0 1 1 1 0 0 0 0 0 1 0 0 0  ; go big
f  0 1 0 0 0 1 0 0 0 0 1 0 1  ; fifty cuff
v  0 1 0 0 0 1 0 0 0 1 1 0 1  ; vivid give
T  0 1 0 0 0 1 1 0 0 0 1 0 0  ; thin ether
D  0 1 0 0 0 1 1 0 0 1 1 0 0  ; this then
s  0 1 0 0 0 1 1 0 0 0 1 0 1  ; source less
z  0 1 0 0 0 1 1 0 0 1 1 0 1  ; zone raise
S  0 1 1 0 0 0 1 0 0 0 1 0 1  ; shy machine
Z  0 1 1 0 0 0 1 0 0 1 1 0 1  ; vision pleasure
m  0 1 0 0 0 1 0 0 0 1 0 1 0  ; murmur dim
n  0 1 0 0 0 1 1 0 0 1 0 1 0  ; no own
N  0 1 1 1 0 0 0 0 0 1 0 1 0  ; sing ink
l  1 1 0 0 0 1 1 0 0 1 1 0 0  ; lily pool
r  1 1 0 0 0 0 1 0 0 1 1 0 0  ; rose rarity
h  0 0 0 0 0 1 0 0 0 0 0 1 0 0  ; hat ahead
w  0 0 1 1 0 0 0 1 0 0 0 0 0  ; we away
y  0 0 1 0 0 0 0 0 0 0 0 0 0  ; yard young
i  1 0 1 0 0 0 0 0 0 1 0 0 0  ; each free
I  1 0 1 0 0 0 0 0 0 0 0 0 0  ; it bin
e  1 0 0 0 0 0 0 0 0 1 0 0 0  ; day made
E  1 0 0 0 0 0 0 0 0 0 0 0 0  ; end then
A  1 0 0 0 0 1 0 0 0 1 0 0 0  ; mad bat
-  1 0 0 1 0 0 0 0 0 0 0 0 0  ; banana collect
^  1 0 0 1 1 0 0 0 0 0 0 0 0  ; bus cut
a  1 0 0 1 1 0 0 0 0 1 0 0 0  ; father cart
u  1 0 1 1 0 0 0 1 1 0 0 0 0  ; ooze too
U  1 0 1 1 0 0 0 1 0 0 0 0 0  ; good foot
o  1 0 0 1 0 0 0 1 1 0 0 0 0  ; bone know
)  1 0 0 1 1 0 0 1 0 0 0 0 0  ; saw all
#  0 0 0 0 0 0 0 0 0 0 0 0 0  ; word-boundary-marker
```

Figure A.2: Phonemes represented as binary vectors according to the Chomsky-Halle feature matrix for the English plural acquisition task.

Curriculum Vitae

Chan-Do Lee was born on March 18, 1953 in Chunchon, Republic of Korea. He attended Seoul National University where he obtained a Bachelor of Arts degree in German Education in 1975. From September 1975 to March 1978 he served in the R.O.K. Army as a squad leader. Following honorary discharge from the army, he taught English and German at several high schools in Korea from 1978 to 1981. He entered the German program at Arizona State University in 1982. Upon completing a Master of Arts degree in German in 1985, he decided to study computer science to satisfy his interests in computers and language. He came to Bloomington and enrolled as a graduate student in the Computer Science Department at Indiana University. During the course of his graduate studies in Computer Science, he was supported financially by the Indiana University Computer Science Department as an Associate Instructor, and by Professors Robert Port and Michael Gasser as a Research Assistant. He also worked as a systems programmer in the department of Speech and Hearing Sciences. He received a Master of Science degree in Computer Science in 1987 and completed the Doctor of Philosophy degree in Computer Science in 1991.