CLASSICISM VS. ELIMINATIVE CONNECTIONISM: LEARNING SIMPLE ARTIFICIAL GRAMMARS WITH BACKPROPAGATION NETWORKS

by

Marius Vilcu
M.Sc., Polytechnic University of Bucharest, 1997

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

In the School
of
Computing Science

SIMON FRASER UNIVERSITY

Summer 2005

APPROVAL

Name: **Marius Vilcu**

Degree: **Doctor of Philosophy**

Title of Thesis: **Classicism vs. Eliminative Connectionism: Learning Simple Artificial Grammars with Backpropagation Networks**

Examining Committee:

Chair: **Dr. Richard T. Vaughan**
Assistant Professor of Computing Science

Dr. Robert F. Hadley
Senior Supervisor
Professor

Dr. Fred Popowich
Supervisor
Professor

Professor Kenneth Aizawa
External Examiner
Chair, Department of Philosophy
Centenary College of Louisiana

Professor F. Jeffrey Pelletier
Internal Examiner
Department of Philosophy and CRC Chair

Date Defended: July 06, 2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

ABSTRACT

One of the central controversies in cognitive science in the last decade is the issue of what kind of mental structure could support complex and systematic behaviour. On one hand, classicists believe that the human mind operates on explicitly structured symbolic representations, where mental representations are characterized by syntactic and semantic structure, and that mental processes operating over those representations are sensitive to their syntactic structure. On the other hand, eliminative connectionists believe the mind is a system composed of simple processing elements (i.e., nodes, units) that resemble biological neurons, and connections that resemble biological synapses between neurons, which can exhibit intelligent behaviour without operating on explicitly structured symbolic representations.

In this thesis I closely look into a number of aspects of this controversy by analysing how several neural network models are able to perform an important high-level cognitive task, such as language understanding. I discovered that although a few of these connectionist models can learn some very specific and simple syntactic patterns, they all have serious problems generalizing their knowledge to novel input, especially when this input is formed with more complex (finite-state) grammars. The major reason for this behaviour is the fact that the training regimen employed by each of those neural networks renders the networks incapable of extracting the sequential structure of the input stimuli.

With regard to the debate between classicism and eliminative connectionism, I argue that neither classicism, nor eliminative connectionism can explain the entire realm

of high-level cognitive processes. Instead, I argue for a paradigm that embodies both classical, traditional symbolic methods and connectionist models.

ACKNOWLEDGEMENTS

I would like to thank my senior supervisor, Dr. Robert F. Hadley for his considerable help, support, and encouragement throughout my PhD program. I am very pleased that I had the opportunity to work under his supervision.

I would also like to thank my supervisor, Dr. Fred Popowich for his helpful comments and suggestions.

Finally, many thanks to my family for their continuous support and love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THESIS OVERVIEW

This thesis addresses one of the major debates in cognitive science, between classicists and eliminative connectionists, regarding the type of mental representations and mental processes that function in the human brain.

As Fodor & Pylyshyn (1988) define it, a *classical* model of the mind is a system that is committed to complex mental representations that are characterized by "combinatorial syntax and semantics" (Fodor & Pylyshyn, 1988), and, at the same time, to mental processes that are sensitive to the syntactic structure of those mental representations. Classical systems are derived from the structure of Turing and von Neumann machines, i.e., they model the human mind by "storing, retrieving, or otherwise operating on structured symbolic expressions" (Fodor & Pylyshyn, 1988). Classicists argue that cognitive architectures need to involve operations on symbols and variables in order to explain and understand human cognition, and that only classical models are committed to that kind of operations (Fodor & Pylyshyn, 1988; Newell, 1980).

Connectionists believe that mental processing resembles the dynamic and graded evolution of activity in a neural network, where each node's activation depends on the connection strengths and activity of its neighbours. In contrast to classicists, many connectionists argue that these neural networks can exhibit intelligent behaviour without involving operations on symbols, and that they represent a "more plausible paradigm for modeling cognition *at all levels of abstraction* than the traditional, algorithmic methods of AI" (Hadley, 1999). Following Fodor & Pylyshyn's 1988 study, some connectionists

adopted a slightly different approach. The so-called implementational connectionists[1] believe that the brain may function like a neural network, but only if this network *implements* symbol-manipulating mechanisms, such as implementing operations over variables, or algebraic rules. However, there are many other connectionists (the so-called eliminative connectionists) who still argue that symbol-manipulation should be entirely *eliminated* from cognitive theories, because classical mechanisms cannot match the flexibility and efficiency of human cognition.

In this thesis I take a closer look at some of the details of the debate between classicism and eliminative connectionism by examining whether eliminative neural networks can exhibit intelligent and systematic behaviour without relying on explicitly structured symbolic representations. In particular, I investigate how factors such as generalization outside the training space (Marcus, 1998, 2001), input representation, and training independence influence several connectionist models (Altmann, 2002; Altmann & Dienes, 1999; Christiansen, Conway, & Curtin, 2001; Christiansen & Curtin, 1999; Elman, 1999; Negishi, 1999; Shultz, 1999; Shultz & Bale, 2001) with regard to their ability to perform a very important cognitive process, namely learning simple and mildly complex artificial grammars. In later sections of this thesis, I analyse the strengths and weaknesses of those models with regard to learning artificial grammars, and explain the reasons for their success or failure.

Each of these models has been proposed as an eliminative connectionist counterexample to one of the best known and controversial studies of grammar learning in humans (Marcus, Vijayan, Bandi Rao, & Vishton, 1999). Marcus *et al.* (1999) found

[1] A term introduced by (Pinker & Prince, 1988).

that, following habituation, 7-month-old infants are able to differentiate between stimuli formed with two different (but very simple) grammars, such as ABA vs. ABB. They concluded that the only possible explanation for the infants' learning behaviour is that they can extract and represent abstract algebraic rules.

The eliminative connectionist models studied in this thesis have also been analysed with regard to their ability to replicate another important study on infants (Gomez & Gerken, 1999). Gomez & Gerken (1999) performed a similar experiment as Marcus *et al.* (1999) on 1-year-old infants. They showed that infants are still able to differentiate between grammatical and ungrammatical test stimuli even when those stimuli are formed with somewhat more complex (finite-state) grammars.

In a recent book, *The Algebraic Mind*, Marcus (2001) evaluates most of these connectionist models with regard to their ability to replicate his experiment on infants (Marcus *et al.*, 1999). In essence, Marcus argues that only the networks that implement some sort of symbol manipulation or an algebraic rule can be successful in replicating his results on infants. He states that none of the purely eliminative connectionist models constitute a serious counterexample to his original claim that eliminative networks are unable to differentiate between two simple grammars. In the following sections of this thesis, I analyse Marcus' arguments with regard to each connectionist model, and investigate whether or not, in my view, he is correct in his reasoning.

The investigation will also contain discussions about the type of input representations employed in each of the connectionist models. Most of the networks discussed in the next chapters use distributed representations. The only exceptions are Altmann & Dienes (1999), and Altmann (2002), which employ localist representations.

As mentioned in the latter sections of this thesis, distributed representations may have several advantages over localist representations, such as biological plausibility, and minimizing the effect of training independence. However, with regard to the networks' ability to differentiate between simple and (especially) more complex grammars, the type of input representation does not seem to have a significant influence.

Other aspects that I will look at are the issues of generalization outside the training space and training independence. These two issues are closely related, and, according to Marcus (1998, 2001), represent some of the major weak points of eliminative connectionist models trained with the backpropagation algorithm. For each of the connectionist models under investigation in this thesis, I analyse the degree in which generalization outside the training space and training independence apply, and how these factors influence the performance of those neural networks.

The discussion of each of these connectionist models will conclude with an in-depth knowledge representation analysis of those networks. The analysis is focused on discovering how networks internally process the input information in order to perform the task of learning a certain simple or more complex grammar. Various knowledge representation analysis techniques are employed, such as principal component analysis and hierarchical clustering, which are performed on both hidden activations and network contributions (products between hidden activations and connection weights).

The thesis concludes with a discussion of the relevance of the various results obtained in my research with regard to the debate between classicism and eliminative connectionism, and the significance of the knowledge representation analysis performed on each connectionist model.

Thesis Outline

In Chapter 1, I describe the issues involving the controversy between classicism and connectionism, and introduce some of the important terms defined in connectionism, such as eliminative vs. implementational connectionism (with regard to the debate between classicism and connectionism), training in neural networks, input representation, training independence, generalization outside the training space, and how each of these factors influences the ability of connectionism models to perform certain tasks.

Chapter 2 details the history of the studies of artificial grammar learning in humans, with focus on Marcus *et al.*'s (1999) and Gomez & Gerken's (1999) studies on infants. It also describes a multitude of analysis techniques that have been employed in my research, such as connection weight analysis, principal component analysis, hierarchical cluster analysis, and hidden activation analysis. I present the characteristics of these techniques, as well as the difference among them, and how they can help in understanding the behaviour of neural networks. At the end of chapter 2 I describe the simulators used in my research, focusing on my own simulator called Virtual Neural Network Simulator (VNNS). I outline the major characteristics of VNNS, and what prompted the introduction of this simulator.

Chapters 3 to 7 describe several connectionist models that attempt to replicate Marcus *et al.*'s (1999) study. Each of these chapters starts with a presentation of the connectionist model (including the results reported by the author(s)), followed by Marcus' evaluation of the model, and my own investigation of both the neural network (with regard to its ability to replicate Marcus *et al.*'s and Gomez & Gerken's studies) and

Marcus' reasoning. Each chapter ends with an in-depth knowledge representation analysis of the model.

In Chapter 3, I analyse Elman's (1999) simple recurrent network. In Chapter 4, I investigate the original Shultz (1999) model, as well as the subsequent Shultz & Bale (1999) network. Chapter 5 is dedicated to two very similar models: Altmann & Dienes (1999), and Altmann (1999). In Chapter 6, I analyse a connectionist model that is the subject of two separate studies: Christiansen & Curtin (1999), and Christiansen *et al.* (2001). Chapter 7 contains the investigation of Negishi's (1999) simple recurrent network.

Chapter 8 lists three attempts to replicate Marcus *et al.*'s (1999) study using *implementational* connectionist models, i.e., the models implement some form of classical mechanism (e.g., variable binding, operations over variables) within a neural network.

Finally, Chapter 9 recapitulates the major findings of my research, and contains a discussion of the significance of these findings, as well as the conclusions that can be drawn from them.

1 INTRODUCTION

1.1 Connectionism

Originally taking its inspiration from the biological neuron and neurological organization, connectionism is a computational paradigm, which attempts to explain human intellectual abilities in terms of artificial neural networks (or, simply, "neural networks", or "connectionist models"). Neural networks are collections of simple processing elements (often called "units', or "nodes"; they correspond to biological neurons), along with weighted interconnections among these elements (the weights measure the strength of connections between units; they correspond to the biological synapses between neurons).

The units in a neural network are usually separated into three classes: input units, hidden units, and output units. Input units receive the information that needs to be processed by the network, and form the network's input layer. Output units provide the results of the processing, and constitute the network's output layer. Hidden units are in between units that help at the processing of the input information. They form the network's hidden layers. Figure 1 displays an example of a simple, feed forward[2] neural network.

[2] In a feed forward network the information travels in only one direction, from the input units to the output units. There are other types of neural networks (e.g., recurrent networks) where the information can travel in both directions.

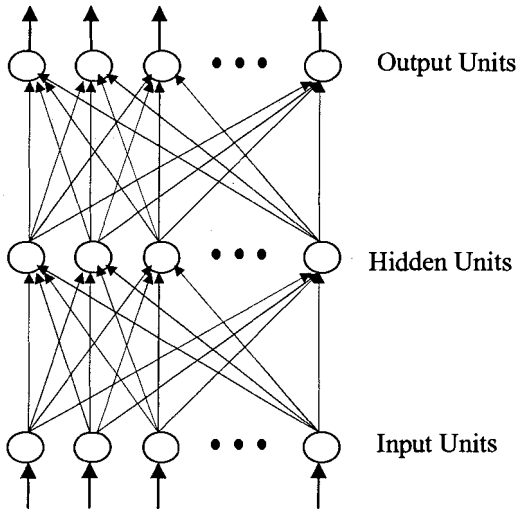

Figure 1 A simple, feed forward neural network (some nodes and connections are not shown)

The input stimuli are presented to the network's input units, each input unit having an activation value that typically represents a feature (or characteristic) of the input stimuli. An input unit sends its activation value to each of the hidden units to which it is connected. Each of these hidden units computes its own activation values depending on the activation values it receives from the input units, and the connection weights (i.e., the strength of connections) towards those input units. The activation values of all these hidden units are then passed on to output units, or to another layer of hidden units. Those hidden units compute their activation values in the same way, and then send them along to their upward neighbours. Eventually, the information propagates all the way through the network, from input units to output units. The activation values of all output units represent the result of the processing by the network of the current input stimulus.

The activation value of each receiving unit is a function of the activation values of all sending units and the connection weights between the receiving unit and all sending

units. There are various ways to compute these activation values. The most common way is to calculate the sum of the products between the activation values of each sending unit and the connection weights between the receiving unit and each sending unit. Sometimes this sum is scaled to a value between 0 and 1, and/or the activation value is set to 0 unless the sum is greater than a given threshold.

1.1.1 Learning in a neural network

When using a neural network to perform a certain task, the central goal is to find a set of connection weights that allows the network to accomplish that task. In order to do that, there are various training methods (or learning algorithms) that can calculate the weights. One of the most widely used training methods is called backpropagation (Rumelhart, Hinton, & Williams, 1986). According to the backpropagation algorithm, a network is presented with a set of training examples consisting of input stimuli and the desired output values for each of those stimuli. For example, if a network is to distinguish between two types of shapes (squares vs. circles), the training set can contain various (say, black and white) pictures depicting squares and circles, along with information regarding the type of the shape for each picture. The network may have two output units: one output unit can indicate whether the input picture depicts a square, whereas the second output unit indicates whether the input shape is a circle. The network may have many input units; for example, each input unit can be associated with the brightness of a pixel (i.e., a small area) in the input image. Each input stimuli is represented by a vector; an element in this vector is associated with a pixel in the image. The size of the input

vectors (which dictates the number of input units[3]) is given by the number of pixels in the input pictures (assuming that all pictures have the same size). Usually, the initial weights of the network to be trained are set to small random values. The training is performed by repeatedly presenting each member of the training set to the network. The elements of each input vector are provided to the input units; the information is then propagated through the network (in a similar manner as described earlier), and the activation values of the output units are compared to the desired (target) values associated to the current input. Then, the connection weights are adjusted in order to minimize the difference between the actual output values and the desired output values. For example, if the current input vector represents an image of a square, the connection weights of the network are changed in order to increase the activation value of the output unit that corresponds to the square-category, and to decrease the activation value of the output unit that corresponds to the circle-category. After many repetitions of all members of the training set, the network may learn to produce the correct output for each input in the training set. It is also possible that the network could generate the correct output for input patterns that were not part of the training set. For example, the network could distinguish between squares and circles in images that were not presented during training.

One of the problems with backpropagation and other learning techniques is that, typically, in order to carry out a specific task, the training of a network may require numerous rounds of weight adjustments, and this may take a long time (hours, days, or even weeks). Also, especially with regard to backpropagation, these kinds of error measurements and weight adjustments are hard to justify biologically. It is unclear

[3] Each vector element corresponds to one input unit; the value of each vector element becomes the activation value of the corresponding input unit.

whether the brain learns by a process like backpropagation, and is unrealistic to think that the human brain needs to employ so many repetitions of the training stimuli that are required by a learning method such as backpropagation.

1.1.2 Simple recurrent networks (SRNs)

The simple recurrent network architecture was introduced by Elman more than a decade ago (Elman, 1990), and has been extensively used to simulate various aspects of cognitive processes, especially language understanding.

The SRN architecture is based on a multi-layer feed forward neural network (see Figure 2). Its main task is prediction: given a temporal sequence of input patterns, the SRN is trained (typically, with backpropagation) to output the most likely pattern that follows the current input stimulus.

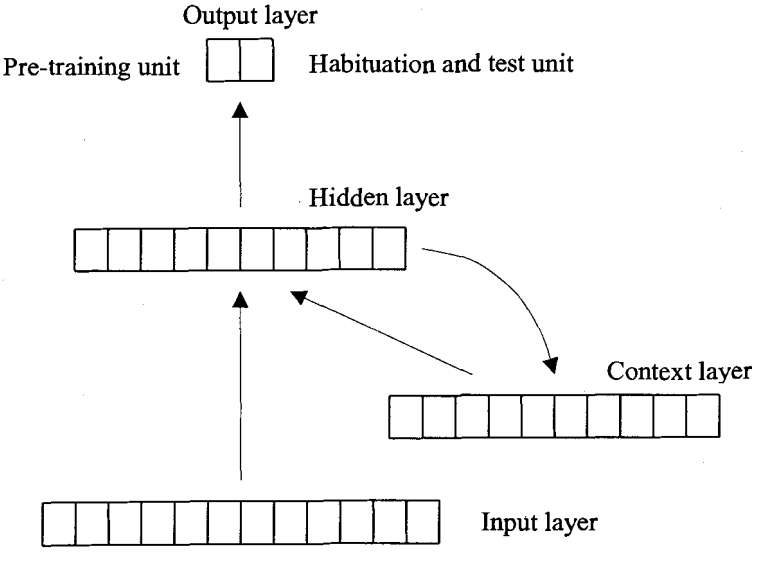

Figure 2 An example of a simple recurrent network (Elman, 1990, 1999; Seidenberg & Elman, 1999)

A typical task for a SRN is the prediction of the next word in a sentence. When training with sentences, each word is presented to the input layer of the network one at a time. At each time step, the network is trained to output the next word in the current sentence. What facilitates this task is the presence of an additional layer, called *context layer*, which has the same size as the hidden layer, and stores the activations of the hidden units (see Figure 2). Each hidden unit has a corresponding context unit to which it is linked through two sets of connections. One set of connections is a one-to-one mapping between each hidden unit and its corresponding context unit (usually this set of connections is not trainable, having a fixed weight of 1). The other set of connections is trainable, and links each context unit to all hidden units. Using this mechanism, at each time step, the contents of the hidden layer are copied to the context layer, and at the next time step the contents of the context layer feed back to the hidden layer. In this way, the context layer works as a short-term memory of the hidden activations, making the network recurrent, and assisting it in its task of predicting the next item in a sequence.

1.1.3 Connectionist representations

Typically, input stimuli are presented to a neural network in two ways: using localist and/or distributed representations. Localist representations use individual nodes to represent an entire concept, whereas distributed representations use a set of nodes to represent a single concept. For example, consider the set of all 26 letters in the Latin alphabet: *a*, *b*, *c*, etc. In a localist representation, each of these letters is represented by a vector of 26 elements [*l1, l2, l3, ..., l26*], each element in the vector corresponding to a letter of the alphabet: *l1* corresponds to letter *a*, *l2* corresponds to *b*, etc. This means that, in order to represent a certain letter of the alphabet, only the element of the vector that

corresponds to that letter is set to 1; all the other elements are 0. For instance, letter *a* is represented by the vector [1, 0, 0, ..., 0], letter *b* is represented by [0, 1, 0, ..., 0], etc. In a distributed representation, each letter may be represented by a set of features (or attributes), such as *is_vowel*, *is_voiced*, *is_nasal*, etc. (Plunkett & Marchman, 1993). In this case, each letter is also represented by a vector, but each vector element corresponds to one of those features[4]. The representation of a letter is determined by the set of features that it contains: all vector elements that correspond to that letter's features will be set to 1; all other elements will be set to 0. For instance, for letter *a*, the feature *is_vowel* will be set to 1, but for letter *b*, the feature *is_vowel* will be set to 0.

Although distributed representations seem more difficult to understand than localist representations, they do exhibit important advantages. First of all, they are usually more biologically plausible. It is unrealistic to think that the human brain employs a single neuron to represent each possible concept. For example, empirical neurological evidence suggests that it is unlikely that there is just one neuron that activates when we think about the concept *dog*. Secondly, distributed representations can better explain various properties of the brain such as "graceful degradation"[5], as demonstrated by several connectionist models (Sejnowski & Rosenberg, 1987; Wood, 1978). In a distributed representation, similar input stimuli will have similar representations. The intrinsic properties of the distributed representation of a stimulus determine its relationships with other stimuli. If a subset of the input nodes disappears, the neural network may still be able to recognize the new input stimuli based on the similarities of

[4] It is possible that a feature may be represented by more than one vector element. For example, a feature such as *stress* (with regard to a syllable within a word) may be represented by two vector elements, in order to indicate primary stress, secondary stress, or no stress (Christiansen, Allen, & Seideberg, 1998).
[5] Sometimes, in case of brain damage, the behavioural performance "gracefully" decreases, rather than exhibiting a complete and sudden breakdown.

their representations to other, known representations. And thirdly, distributed representations typically need fewer nodes than localist representations do. For example, a network needs 26 input nodes to locally represent all 26 letters of the alphabet, whereas, based on Plunkett & Marchman's (1999) feature notation, a network may only need 6 input nodes to represent each of the 26 letters.

1.1.3.1 Superposition catastrophe

In spite of their advantages over localist representations, many consider that distributed representations have their own problems. For instance, Marcus (2001) considers that distributed representations may be subject to a problem called *superposition catastrophe* (Hummel & Holyoak, 1993; von der Malsburg, 1981). This can occur when the representations of various input or output entities overlap. For example, based on the previous example with the Latin alphabet, considering a distributed representation, if letter *a* is represented by the vector [1, 0, 1, 0, 0, 1], letter *b* by [0, 1, 0, 1, 1, 0], letter *c* by [0, 1, 1, 1, 0, 0], and letter *d* by [1, 0, 0, 0, 1, 1], then the network will not be able to distinguish the simultaneous activations of letters *a* and *b*, from letters *c* and *d* (both combinations are represented as [1, 1, 1, 1, 1, 1]).

Typically, superposition catastrophe may occur when distributed representations are used on the output layer of a network (in general, entities are presented individually to the input layer). For example, in a simple recurrent network, the usual task is to predict the most probable items that can follow the current input item in a temporal sequence (Elman, 1990). For instance, if the next word in the current sentence is a noun, the network may activate all output units associated with the features of all possible nouns. If

those features overlap with (say) the verb features, it may be impossible to unambiguously distinguish the network's response.

I agree that, theoretically, superposition catastrophe can happen. However, I believe there are many ways to overcome it, and none of the connectionist models discussed in this thesis exhibits this "problem". One way to overcome superposition catastrophe is to employ a sufficiently rich and well-defined distributed representation that minimizes the amount of overlap among features. Although I agree that, for instance, some of the noun and verb features may overlap[6], for a sufficiently rich distributed representation, many of those features do not overlap (Plunkett & Marchman, 1993). Another way to overcome this problem is to use localist representations on the output layer (Christiansen & Curtin, 1999; Elman, 1999).

1.2 Classicism vs. Connectionism

One of the central controversies in cognitive science in the recent years refers to the type of cognitive architectures that can model the human mind, with regard to explaining both the kind of mental structures that support complex and systematic behaviour, as well as the mental processes that operate over those structures.

The controversy started with a very influential paper by Fodor & Pylyshyn (1988). They postulate that human mind exhibits a so-called "language of thought", where mental representations have "combinatorial syntactic and semantic structure" (Fodor & Pylyshyn, 1988). They argue that mental representations are structurally complex and have syntactic constituents that can be either structurally complex or

[6] For example, if the representation uses a feature like "is_animate", this may be included in both nouns and verbs.

structurally atomic. Also, the semantic content of a mental representation depends on the semantic contents of its syntactic constituents. This entails that, according to Fodor & Pylyshyn, the mind exhibits compositionality and "systematicity", i.e., the ability to perform a certain mental process implies the ability to perform other semantically related mental processes. For example, English speakers who understand *John loves Mary* can also understand *Mary loves John*. In the classical point of view, this ability can easily be explained by assuming that English speakers represent the constituents *John*, *loves*, and *Mary* of the sentence *John loves Mary*, and compute its meaning from the meanings of those constituents. Based on this assumption, the understanding of a novel sentence like *Mary loves John* can be explained by the simple fact that it is a new instance of the same symbolic process. In Fodor & Pylyshyn's view, only classical models are committed to the "language of thought" theory, where mental representations are characterized by combinatorial syntax and semantics, and where mental processes are sensitive to the structure of those representations. They argue that connectionist systems cannot represent genuine models of the human mind, because they acknowledge "neither syntactic nor semantic structure in mental representations" (Fodor & Pylyshyn, 1988). According to Fodor & Pylyshyn, unless connectionism is used as an implementational theory[7], it cannot explain why systematicity is found so pervasively in human cognition. Fodor & McLaughlin (Fodor & McLaughlin, 1990) also noted that connectionism does not guarantee systematicity, because, although connectionist models can be trained to be systematic, they can also be trained, for example, to understand *John loves Mary*, but not understand *Mary loves John*.

[7] That is, implementation of classical mechanisms.

Many connectionists disputed Fodor & Pylyshyn's study as being simplistic, providing "no succinct and precise characterization of systematicity" (Niklasson & van Gelder, 1994), and minimizing the way in which connectionist representations possess internal structure, as noted in (Chalmers, 1990), and (Smolensky, 1988). Also, as mentioned in (Aizawa, 1997), (Hadley, 1997), and (Matthews, 1997), classical architectures have problems at explaining systematicity as well. For example, there are classical models that can be programmed to understand *John loves Mary*, but not understand *Mary loves John*. The opinion is that neither connectionist mechanisms alone, nor classical theories alone, can explain pervasive systematicity. In both architectures, further assumptions about the nature of the processing may be needed in order to ensure that *Mary loves John* is processed as well.

However, following Fodor & Pylyshyn's controversial paper, many connectionists started adopting an approach that supports both classical and connectionist points of view. The so-called *implementational* connectionists believe that the brain is indeed a symbolic processor, but at a more abstract level, and that symbolic processing is *implemented* in a structure similar to a neural network. Implementational connectionists attempt to discover a way for classical processing to be carried out by neural networks.

There are many other connectionists (the so-called *eliminative* connectionists) who believe that the mind does not operate on explicitly structured symbolic representations. According to eliminative connectionists, symbols should be *eliminated* from the study of cognitive processes. They argue that classical mechanisms cannot match the flexibility and efficiency of human cognition, and cannot explain several features of human intelligence, such as graceful degradation.

It is noteworthy that, in a series of studies, Marcus (1998, 1999, 2001) defines a slightly different conception of classicism. He argues that symbols, variables, and operations over these variables represent the underlying features of the human mind, and they should constitute the basis for all scientific theories regarding cognition. In his view, in order to explain how humans perform various cognitive processes, such as extending universals to arbitrary items (for example, if humans know that *all birds can fly*, and that *Tweety* is a bird, they infer that *Tweety can fly*), or generalization to novel instances of variables (for example, if humans know that *a rose is a rose*, *a tulip is a tulip*, etc, they predict that *blicket* is the continuation to *a blicket is a*...), cognitive architectures need to implement operations over variables.

With regard to the debate between classicism and eliminative connectionism that is analysed in this thesis, Marcus' concept of classicism is considered, rather than Fodor & Pylyshyn's.

1.3 Training Independence

As mentioned earlier, backpropagation is a training algorithm that is frequently used in today's neural networks. Figure 3 displays the backpropagation training rule, according to (Rumelhart *et al.*, 1986).

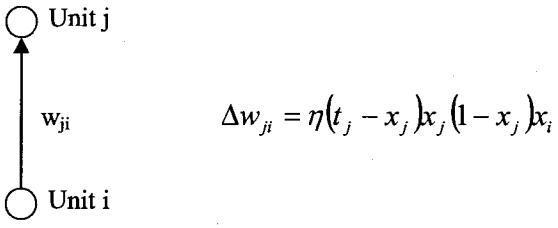

Figure 3 The backpropagation rule (Rumelhart *et al.*, 1986)

w_{ji} represents the connection weight between the receiving unit j and sending unit i, η is the learning rate (typically, a constant value between 0 and 1), t_j represents the target value for unit j, and x_i, x_j are the activation values of units i, and j, respectively.

According to this backpropagation rule, the weights feeding a given node are trained independently from the weights feeding any other node on the same layer: Δw_{ji} only depends on the properties of units i and j. Marcus (1998, 2001) calls this *output independence*, because the set of weights connecting one unit to its input units is entirely independent from the set of weights feeding all other units on the same layer.

At the same time, Marcus argues that, in many connectionist models, there are input nodes that are not correlated with the network's output for the entire duration of training. For example, if a neural network is trained with a subset of letters of the Latin alphabet (a, b, c, …, z) using localist representation (each letter corresponds to a single input node), there will be nodes that will have never been activated during training (i.e., their input activation will be zero for the entire duration of training). Based on the backpropagation rule (see Figure 3), since the weight changes are directly proportional to

the activation of the sending units (x_i), the weights connecting those input units to their neighbours will never be updated. Therefore, training will not have any effect on those connection weights, i.e., the network is unlikely to "extend learning from trained input nodes to untrained input nodes" (Marcus, 1998). Marcus calls this *input independence*, and claims that training independence (the combination of input and output independence) represents a major limitation of eliminative neural networks trained with backpropagation.

Although Marcus argues that training independence equally occurs in connectionist models using both distributed and localist representations, as specified later in this thesis, there are ways to minimize its effects in networks employing distributed representations. In essence, it is possible to define sufficiently rich distributed representations that can allow the training of all input and/or output nodes. According to Marcus' definition of input independence, as long as all input nodes are trained, input independence cannot occur.

1.4 Generalization Outside the Training Space

Marcus (1998, 1999, 2001) claims that eliminative connectionist models cannot perform those cognitive functions that require generalization outside the space of the training examples.

In geometric terms, Marcus (1998, 2001) defines the *input space* as the set of all possible input vectors, the *training set* as the set of input vectors that are used during training, and the *training space* as the area of the input space where the training set

resides. Subsequently, Marcus argues that any input vector that contains features that a connectionist model has not been trained on lies outside the training space.

This definition is ambiguous, however, because it appears compatible with two differing notions of "outside the training space". In the first notion, any input value that corresponds to a feature that was entirely untrained is a value that lies outside the training space. This corresponds to a case where an input node is never activated during training. In the second notion, Marcus states that the training space is delimited by the values of the features that appear in the training set. This corresponds to a case where, if an input node is trained with values between (say) 0.1 and 0.9, and then, during the test phase, it is presented with a value of 1.2, this value will be outside the training space. But, according to the first notion, this value would lie within the training space, because the node was activated during training. In his discussions of various connectionist models regarding their ability to generalize outside the space of the training examples, Marcus (1998, 2001) uses the first notion, and argues that any neural network trained with the backpropagation algorithm (or any variant of it) is not able to generalize outside the training space. Marcus maintains that the reason for this behaviour is that essential aspects of the backpropagation algorithm (e.g., the training independence) preclude the network from generalizing to nodes that have not been specifically trained.

The ability of connectionist models to generalize beyond the set of training examples is very important, especially when networks need to simulate various cognitive processes, like those in the sentence-prediction model (Marcus, 1998). Marcus showed that when humans are presented with a series of simple sentences such as "a rose is a rose", "a lily is a lily", "a tulip is a tulip", they easily predict that "blicket" is the

continuation to the sentence "a blicket is a ..." (Marcus, 1998). If eliminative connectionist models were able to perform the same kind of generalization beyond the space of the training examples, this would support the theory according to which these networks are genuine models of the human mind. Therefore, in this thesis I investigate the ability of several connectionist models to generalize outside the training space within the task of learning simple or more complex grammars. I analyse if and how Marcus' definition of "generalization outside the training" applies to each model, and whether these models are able to generalize to stimuli that are not part of the training set.

2 FURTHER PRELIMINARIES AND RELEVANT BACKGROUND

2.1 Artificial Grammar Learning in Humans

Artificial grammar learning has been extensively studied in humans. Typically, humans are exposed to a set of stimuli generated by a finite-state grammar (see Figure 4). After the acquisition of the training examples, participants are presented with a novel set of stimuli, and they are asked to discriminate between new valid (i.e., grammatical, familiar) examples and invalid (i.e., ungrammatical, unfamiliar) ones. It has been shown that humans can learn mildly complex artificial grammars (Altmann, Dienes, & Goode, 1995; Mathews, Buss, Stanley, Blanchard-Fields, Cho, & Druhan, 1989; Reber, 1967). The learning can be retained over as much as two years (Allen & Reber, 1980), and can occur in both infants (Gomez & Gerken, 1996; Marcus *et al.*, 1999), and adults (Gomez, Gerken, & Schvaneveldt, 2000; Meulemans & Van Der Linden, 1997).

A very important aspect of artificial grammar learning in humans is generalization to a new vocabulary. This is important because it can demonstrate that humans are not only able to acquire specific sequential dependencies, but are also abstracting the sequential structure. Many studies have shown that humans (both adults and infants as young as 7 months of age) are able to generalize to a new vocabulary (Altmann *et al.*, 1995; Brooks & Vokey, 1991; Gomez & Gerken, 1999; Marcus *et al.*, 1999; Mathews *et al.*, 1989; Reber, 1967). In this thesis I focus on the ability of several connectionist

models to replicate two of these studies: Marcus *et al.* (1999) and Gomez & Gerken (1999).

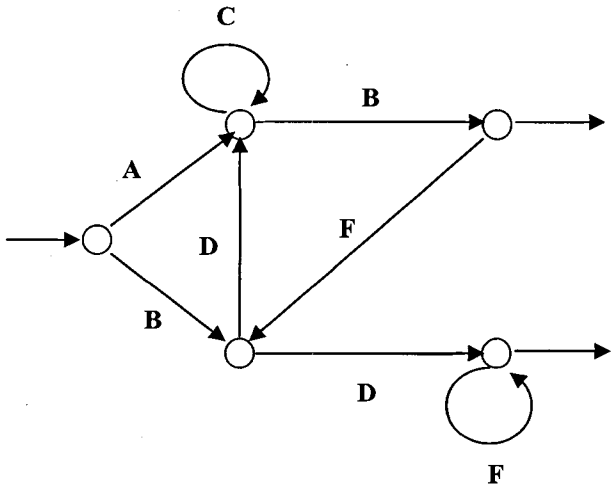

Figure 4 **A finite-state grammar. Grammatical strings are generated by traversing the links from left to right, starting at the leftmost state, e.g.: ACB, BDBFD.**

2.1.1 Marcus *et al.* (1999)

Marcus *et al.* (1999) studied the ability of 7-month-old infants to differentiate between sequences ("sentences") of two-letter syllables generated by very simple grammars, such as ABA, ABB, and AAB. Marcus *et al.* performed three different experiments: in the first two experiments they used the grammars ABA and ABB, whereas in the third experiment they employed the grammars AAB and ABB[8]. Besides the difference in grammars, the three experiments differed in input stimuli. Experiment 2 had a slightly changed input corpus from experiment 1, to compensate for specific phonetic characteristics of the training and test patterns that occurred in the first

[8] Examples of ABA, ABB, and AAB sentences: *ga ti ga, li na li, ta ti ta, li gi gi, ni la la, le le di, wi wi li.*

experiment[9]. In experiment 3 the input corpus was the same as in experiment 2, but the grammars were different (ABB vs. AAB), to balance reduplications in the input stimuli (in the third experiment both grammars contain one duplicated element). Sixteen infants were randomly assigned to either an ABA condition or an ABB condition (in experiments 1 and 2), or to either ABB or AAB (in experiment 3). They were familiarized with 16 sentences generated with one grammar for a period of two minutes. After this habituation period, infants were presented with four novel sequences of syllables constructed from both the familiar (training) grammar, and an unfamiliar grammar (2 sentences with the familiar grammar, and 2 with the unfamiliar one). Marcus *et al.* (1999) discovered that infants showed an attentional preference for sentences that were constructed from the novel (unfamiliar) grammar. Marcus *et al.* argue that the only explanation for such behaviour is that infants possess a rule-learning mechanism that enables them to "represent, extract, and generalize abstract algebraic rules" (Marcus *et al.*, 1999). Also, they claim that this rule-learning mechanism is not available to eliminative connectionist models, because these models "can simulate knowledge of grammatical rules only by being trained on all items to which they apply; consequently, such mechanisms cannot account for how humans generalize rules to new items that do not overlap with the items that appeared in training"(Marcus *et al.*, 1999). In other words, Marcus *et al.* believe that, because this simple discrimination task performed by infants requires them to generalize outside the training space, and because, according to Marcus *et al.*, eliminative

[9] Many A syllables in experiment 1 started with a voiced consonant, whereas the B syllables started with an unvoiced consonant. Therefore, infants who were familiarized with ABA patterns (voiced-unvoiced-voiced structure) could have been surprised by the new voiced-unvoiced-unvoiced combinations of the unfamiliar ABB patterns.

connectionist models cannot generalize outside the training space, these models are not able to perform the same task as the infants'.

One may argue that the grammars employed in this study are too simple to motivate such strong statements from Marcus *et al.* (1999). However, as I mentioned above, Marcus *et al.* run their experiments on very young infants, who are "on the cusp of language learning" (Marcus *et al.*, 1999). I believe that Marcus *et al.*'s reason for using simple grammars was to demonstrate that even these young infants are able to learn simplified versions of algebraic rules. They assume that if they can prove that humans are able to learn artificial grammars at that young age, it is because they "extract abstract algebra-like rules" (Marcus *et al.*, 1999), and that statistical mechanisms alone are not sufficient to explain their behaviour even when using such simple grammars.

2.1.2 Gomez & Gerken (1999)

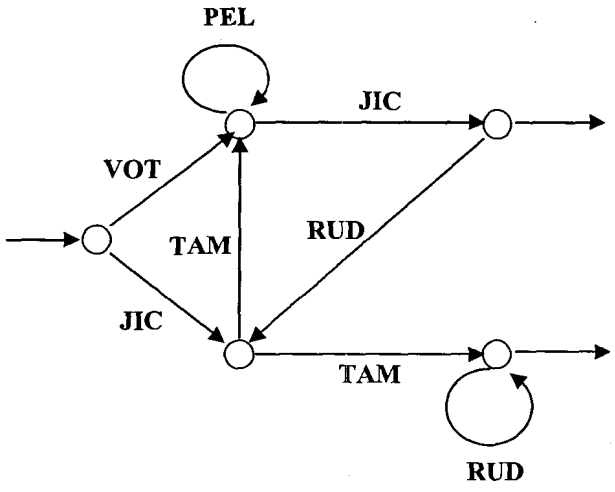

Figure 5 **Gomez & Gerken's (1999) finite-state grammar. Grammatical sentences are generated by traversing the links from the left-most state to right-most states. Examples of grammatical sentences: VOT PEL JIC, JIC TAM JIC RUD TAM RUD.**

Using the finite-state grammar shown in Figure 5, Gomez & Gerken (1999) performed four different experiments with 1-year-old infants. Similarly to Marcus *et al.* (1999), Gomez & Gerken familiarized 16 infants with ten 3-to-6-word sentences generated by a finite-state grammar, for a period of 2 minutes.

Following the habituation period, Gomez & Gerken performed four different experiments:

1. In experiment 1, infants were presented with 10 novel test sentences generated from the familiar grammar, and another 10 novel test sentences generated from the grammatical test sentences by switching the first and last words (i.e., 10 novel test sentences have illegal endpoints). For example, from the grammatical sentence VOT PEL PEL PEL JIC, the ungrammatical sentence JIC PEL PEL PEL VOT was generated. This procedure was done in order to avoid the possibility of infants discriminating between grammatical and ungrammatical test sentences solely based on the sentence length or word frequency.

2. In experiment 2, infants were tested with the same set of grammatical sentences as in experiment 1, but the ungrammatical test sentences had internal grammatical violations. The ungrammatical test sentences started and ended with grammatical words, but the order of the internal words was ungrammatical. For example, the sentence VOT TAM PEL RUD JIC begins and ends with grammatical words, but the internal transitions VOT TAM, PEL RUD, and RUD JIC are ungrammatical. Similar to experiment 1, the ungrammatical test sentences were matched with the grammatical ones in terms of length and word frequency.

3. In experiment 3, infants were tested with 10 novel grammatical sentences, and another 10 sentences generated with a different grammar (see Figure 6). The grammatical and ungrammatical test sentences begin and end with the same words, but the internal transitions are different. However, the grammatical and ungrammatical test sentences are matched in terms of length and word frequency.

4. In experiment 4, infants were tested with the same sets of grammatical and ungrammatical structures as in experiment 3, but both types of test sentences were constructed with a different vocabulary of words (VOT, PEL, JIC, RUD, TAM) that the one used during the habituation period (JED, FIM, TUP, DAK, SOG).

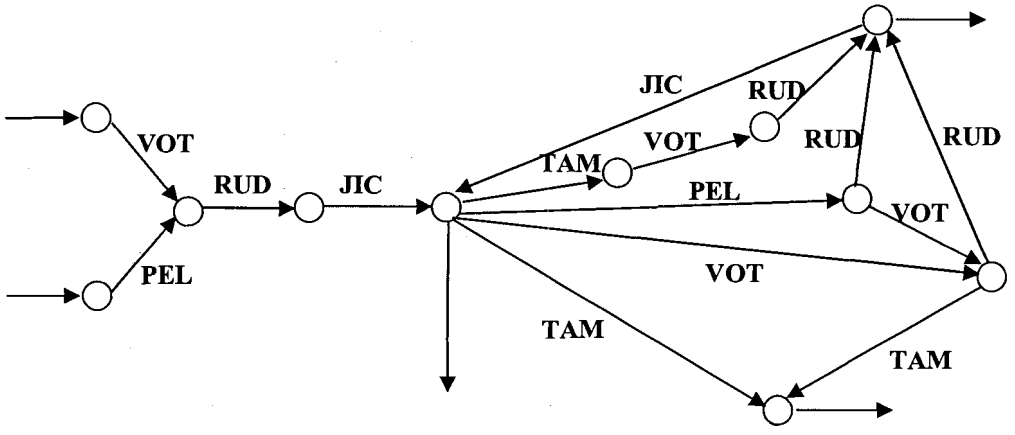

Figure 6 Finite-state grammar used to generate sentences in Gomez & Gerken's experiments 3 and 4. Grammatical sentences are generated by traversing the links from left to right. Examples of grammatical sentences: VOT RUD JIC, PEL RUD JIC VOT RUD, VOT RUD JIC TAM VOT RUD.

In all four experiments, Gomez & Gerken discovered that infants were able to discriminate between the test sentences that were generated from the familiar grammar, and the test sentences that were generated from the unfamiliar grammar or violated the familiar grammar. Gomez & Gerken claim that infants show "remarkable abstraction

abilities" (Gomez & Gerken, 1999), by extracting information regarding the sequential structure of the input sentences, even when the test sentences are generated in a completely new vocabulary.

During my investigation of various connectionist models discussed in this thesis, I have been mostly interested in analysing the ability of those networks to perform experiment 4 of Gomez & Gerken (1999). Only experiment 4 requires participants to generalize beyond the training examples, in a similar way to that in the Marcus *et al.*'s (1999) study, and tests the capacity of those participants to abstract the sequential structure of the input sentences.

2.2 Knowledge Representation Analysis

There are various ways to analyse and explain the behaviour of neural networks. Throughout this thesis I employ hierarchical cluster analysis, principal component analysis (PCA), connection weight analysis, and hidden unit activation analysis.

2.2.1 Hierarchical Cluster Analysis

Hierarchical clustering is a statistical method for identifying homogeneous subgroups (or clusters) of cases in a data set by creating a hierarchical cluster tree. Typically, cluster analysis involves three steps:

1. Finding the similarity or dissimilarity between every pair of objects in the data set. This is usually done by calculating the Euclidian distance between every pair of objects, and generating a so-called distance or dissimilarity matrix.

2. Grouping the objects into a binary, hierarchical cluster tree. Based on the dissimilarity matrix computed at step 1, the objects that are close together are

paired into binary clusters. Then, the newly formed clusters are grouped into larger clusters until a hierarchical tree is formed with all objects in the data set. The tree diagram used to represent the results of a cluster analysis is called *dendrogram.*

3. Determining the clusters generated at step 2 by detecting the natural groupings in the hierarchical tree or by forcing the grouping into an arbitrary number of clusters.

Cluster analysis has been extensively used in the study of neural networks, especially with regard to the analysis of how networks form internal representations (Elman, 1990). In this thesis, cluster analysis is performed on the hidden activation vectors formed by the test items. This analysis is important because it can show the relationships among various internal representations of input stimuli, which may help at explaining the networks' reaction to those stimuli. For example, if a network's internal representations generated by certain input vectors cluster (or group) according to the hierarchical tree, one may conclude that there is a common pattern of network activity for those vectors (i.e., those input vectors are similar from the network's point of view). Alternatively, if several internal representations do not group, this may indicate that the network detected dissimilarities among the input vectors that generated those internal representations.

2.2.2 Principal Component Analysis (PCA)

PCA (also called Karhunen-Loeve transform, or Hotelling transform) is a statistical method for reducing the dimensionality of a data set, while retaining as much

information as possible, by computing a compact and optimal description of the data (Flury, 1988; Jolliffe, 1986).

PCA involves a mathematical technique that transforms a set of m correlated variables into a set of n uncorrelated variables, where, typically, $n << m$. The goal of PCA is to find a number of independent components[10] (also known as *principal* components, or *eigenvectors*) that can explain the maximum amount of variation in the data set. The first principal component is the combination of variables that accounts for the greatest amount of variation in the data. The second principal component explains the next largest amount of variation that cannot be explained by the first component, etc.

Jolliffe (1986) has proved that the representation generated by PCA is an optimal linear dimension reduction technique. The reduction in dimension is important because, besides decreasing the computational overhead and the noise, it allows a better visualization of the data, especially when n is small (less than 3).

If the data set contains a significant amount of variation, the number of principal components generated by PCA can be high. However, there are methods to eliminate those principal components that do not account for much of the variance, such as the *scree* test[11] (Cattell, 1966), or by keeping only those components whose eigenvalues are greater than the average eigenvalue (Shultz & Bale, 2001).

For several connectionist models that are analysed in this thesis, PCA reveals more than three principal components, even after applying the reduction techniques discussed above. In those cases, it is difficult to display all components on the same

[10] There can be as many components as there are variables.
[11] The scree test eliminates those components whose eigenvalues level out in the eigenvalue graph (i.e., they are close enough to zero that they can be ignored).

graph. However, in each of those cases, I discovered that the first two principal components typically reveal the most important characteristics of the data. Also, the projections on the third and fourth components may reveal interesting characteristics (Altmann, 2002; Altmann & Dienes, 1999; Elman, 1999). Other combinations of principal components (e.g., the first and third components, the second and fourth components, etc) do not expose any new important features of the data. Based on these findings, in each of those situations, I have chosen to display a set of 2-dimensional graphs, where each graph contains the projections on only two principal components: the first graph projects the results of PCA on the first two principal components, whereas the second graph projects the results on the third and fourth components.

PCA has been widely used in the analysis of neural networks in order to reduce the variability of the data being analysed. PCA is usually applied to the hidden activations of a neural network (Elman, 1989), or to the network contributions (Shultz & Elman, 1994; Shultz, Oshima-Takane, & Takane, 1995). As defined in (Sanger, 1989), contributions are the products between the hidden unit activations and the connection weights between the hidden units and the output units. Contributions are valuable in network analysis because they account for both hidden activations and connection weights, and can balance any extreme values for either connection weights or hidden activations. Contributions are also useful in the analysis of networks where hidden units are organized in such a way that the study of hidden activations alone is problematic. For example, in Shultz' model (1999), the cascade-correlation algorithm generates one unit on each hidden layer. The knowledge is distributed across many hidden layers, and there are connections that bypass some of the hidden units (the so-called *cross-connections*). In

these situations, the analysis of network contributions may provide a more accurate picture regarding a network's behaviour.

2.2.3 Connection Weight Analysis

Not as widely used as other analyzing techniques, connection weight analysis may sometimes provide valuable information. Typically, the connection weights linking the output nodes to the units on the last hidden layer are studied, and certain roles for output units may be discovered. For example, by performing this kind of analysis, Shultz & Bale (2001) discovered that the connection weights going into certain output nodes are similar to the connection weights going into another set of output nodes. This fact made them argue that those two sets of nodes represent the same entity[12].

2.2.4 Hidden Unit Activation Analysis

The activations of hidden units may provide very valuable information with regard to how the networks internally represent the input stimuli. Because the number of hidden units is usually high (greater than 3), hidden activations are typically subjected to hierarchical cluster analysis (Elman, 1990), or PCA (Elman, 1989). By examining the hidden activations, one can discover important characteristics of a network, such as whether it forms clusters of representations for certain input stimuli (Elman, 1990), or whether the hidden units are sensitive to certain characteristics of the stimuli (Shultz & Bale, 2001).

[12] However, as argued later in the thesis, I believe that Shultz & Bale's argument is wrong.

2.3 Neural Network Simulators

Most of the network simulations presented in this thesis have been performed with my own simulator called Virtual Neural Network Simulator (VNNS). The results were verified using Carnegie Mellon University's PDP++ 2.2 simulator (O'Reilly & Munakata, 2000) and Elman's TLearn software (Plunkett & Elman, 1997). Because of its very specific structure (i.e., simple recurrent network without hidden/context layers), the only instance where I have not used my own simulator was during the investigation of Negishi's model (1999). In that particular case, I have used Negishi's own simulator (Negishi, 1999).

2.3.1 Virtual Neural Network Simulator (VNNS)

This simulator is a C++, Windows-based system, which performs backpropagation training on both feed-forward neural networks, and simple recurrent networks. Additionally, VNNS implements the cascade-correlation algorithm (Fahlman & Lebiere, 1990).

VNNS is highly parameterized, allowing for numerous alterations of training parameters such as learning rate, momentum, number of training epochs, distribution of initial weights (minimum and maximum values), initial activation of context units (for simple recurrent networks), etc., using a simple graphical interface. The input data is stored in external text files, one input item per line. VNNS reads the training and test data from those text files, and saves the results of training and test (including the activation values of all units after each training and test stimulus, the strength of all connection weights following training, output errors) onto text files. The names of the input and output files are also parameterized.

Another important feature of VNNS is that it allows for running training simulations for any number of networks, in a sequential manner (i.e., batch simulations), without user intervention. The instructions for running the batch simulations are defined using a simple scripting language, and included in a text file. This text file is then loaded and executed by VNNS.

For example, this is the script for running 8 different simulations of Elman's model (1999):

```
loadNet ("elman.net")
for 8 cycles
   reset
   loadTrPattern ("aba_train50.pat")
   loadTcPattern ("aba_teach50.pat")
   train ("6")
   loadTrPattern ("aba_train96.pat")
   loadTcPattern ("aba_teach96.pat")
   train ("347")
   loadTestIn ("aba_test.pat")
   loadTestOut ("aba_testout.pat")
   testAndSave ("output.txt")
endfor
```

where:

loadNet: an instruction for loading the network's architecture from an external file. In this case, a file called "elman.net" contains the specifications of Elman's simple recurrent network (the number of input units, the number of hidden/activation units, the number of output units, learning rate, momentum, initial weight limits).

for 8 cycles: an instruction for running 8 different simulations (i.e., training of 8 different networks).

reset: initializes the network (connection weights, activations).

loadTrPattern: an instruction for loading the input training patterns from an external file. In the case of Elman's simulation, this instruction is called twice. Firstly, a file called "aba_train50.pat" containing 50,000 syllables (one syllable per line) used in the preliminary training phase is loaded. Following the pre-training phase, a file called "aba_train96.pat" containing the ABA training patterns is loaded.

loadTcPattern: an instruction for loading the output (target) vectors for the training patterns from an external file. In this case, for the preliminary training phase, a file called "aba_teach50.pat" containing all 50,000 target vectors (one vector per line) used in the pre-training phase is loaded. Following the pre-training phase, a file called "aba_teach96.pat" containing the output (target) vectors for all ABA training patterns.

train: instructs VNNS to train the network for a number of iterations. In Elman's case, it is called twice: once for the pre-training phase (6 iterations), and once for the training phase (347 iterations).

loadTestIn: an instructions for loading the input test patterns from an external file. In this case, a file called "aba_test.pat" contains Elman's input test patterns.

loadTestOut: an instruction for loading the output (target) test patterns from an external file. In this case, a file called "aba_testout.pat" contains Elman's output test patterns.

testAndSave: instructs VNNS to run the test phase and then save the network's parameters (it saves the hidden activations for all training and test patterns, connection weights, output vectors).

endfor: marks the end of the *for* cycle.

I have used VNNS to simulate most of the simple recurrent networks discussed in this thesis, such as (Altmann, 2002; Altmann & Dienes, 1999; Christiansen & Curtin, 1999; Elman, 1999; Shultz, 1999; Shultz & Bale, 2001).

What prompted the creation of this simulator was the inability of the PDP++ software to deal with specific training techniques, such as the one employed by Elman (1999)[13]. Also, PDP++ is not able to simulate the cascade-correlation algorithm, which is used in two of the models discussed in this thesis: Shultz (1999), and Shultz & Bale (2001). However, whenever a connectionist model that I have been investigating could have been simulated by PDP++, the results reported by VNNS were verified against PDP++. There have been no statistically significant differences between the results reported by VNNS and those reported by PDP++.

[13] As mentioned later in the thesis, Elman (1999) does not update the connection weights for all training patterns.

3 ELMAN'S MODEL

Since the 1998 Marcus study on eliminative connectionist networks and their apparent inability to generalize outside the training space, Elman has tried to prove Marcus wrong in a series of studies (Elman, 1998, 1999, 2001; Seidenberg & Elman, 1999).

3.1 Network Architecture and Preliminaries

In their original experiment on the ability of infants to differentiate between simple grammars, Marcus *et al.* (1999) use novel stimuli during testing, and this prompts them to reiterate the theory that connectionist models will not be able to replicate their results (because, according to Marcus, connectionist models are not able to generalize outside the space of the training examples). However, shortly afterwards, Elman (Elman, 1999; Seidenberg & Elman, 1999) presents a model that he believes represents a counterexample to Marcus *et al.*'s (1999) claims: a simple recurrent network (SRN) with 12 input units, 10 hidden units, and 2 output units (see Figure 2).

When building his model, Elman makes an important assumption: he believes that, before their participation in Marcus *et al.*'s (1999) study, infants do experience familiarization to millions of words, and are "aware of the surrounding language's phonotactic regularities" (Elman, 1999). Based on this assumption, Elman argues that if connectionist models are to have any chance of success in replicating Marcus *et al.*'s results, they need to be presented with similar familiarization data. In Elman's view, this

can be accomplished by a preliminary training process during which the network is exposed to a large number of words, intended to capture infants' prior experience.

3.2 Input Representation

The 12 input units of the network encode each word (2-letter syllable) that is presented to the network, based on the feature representation developed by Plunkett & Marchman (1993) for vowels and consonants (see Figure 7). The first 6 input units encode the consonant being presented, while the other 6 input units encode the vowel. Plunkett & Marchman represent each phoneme as a 6-bit vector having the following features:

- Type: Consonant or vowel (1 bit)
- Voicing: voiced or un-voiced (1 bit)
- Manner (2 bits)
- Place (2 bits)

Examples of several input words based on Plunkett & Marchman's (1993) notation are displayed in Figure 7. If a phoneme has a certain feature, the vector elements associated to that feature are set to +1. Otherwise, they are set to –1.

3.3 Training Procedure

For the task of replicating Marcus *et al.*'s (1999) experiments, Elman trains the network in two phases. In the first phase (called preliminary training, or pre-training), he generates a corpus of 120 two-letter syllables (e.g., *ba*, *po*, *je*, etc; the first letter is a consonant and the second letter is a vowel), and then creates a sequence of 50,000

syllables taken from the 120-syllable corpus. Each syllable from the sequence is presented to the input layer of the network, one at a time, and Elman pre-trains the network to distinguish whether two consecutive syllables in the 50,000-word sequence are identical or not. In this pre-training phase he only employs one of the two output units, the target outputs being 1 (when the current syllable is identical to the previous one), and 0 (when the syllables differ). With this process Elman intends to simulate the experience that infants have before participating in Marcus *et al.*'s study.

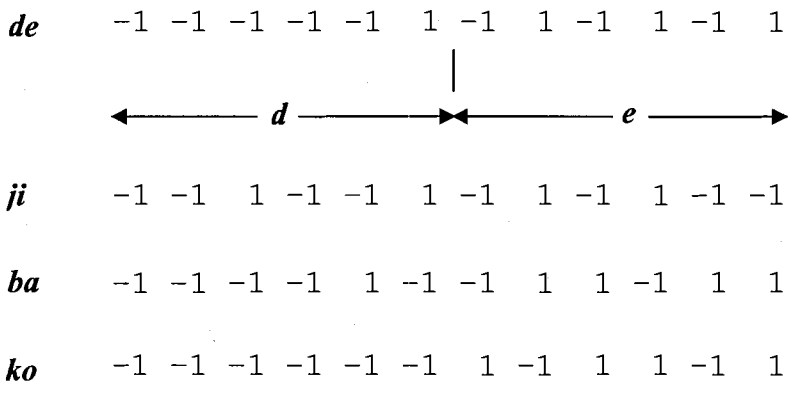

Figure 7 Plunkett & Marchman's (1993) notation used in Elman's model.

Following the pre-training phase, using a subset of 8 syllables from the 120-syllable corpus, Elman generates 32 sequences of syllables ("sentences") having the form ABA and ABB (16 ABA and 16 ABB sentences). For example, a few of the sentences used during training are: *le di le*, *wi je wi*, *le di di*, *wi je je*.

Elman presents each of these sentences to the network, in random order, one syllable at a time, and attempts to train the network to distinguish between the two grammars. In this phase, Elman employs only the second output unit (the one *not* used

during pre-training), the target outputs being 0 in the case of ABA patterns, and 1 in the case of ABB patterns. The actual training (change of the connection weights) is done only after the last syllable in the sentence is presented. No training occurs immediately following the first two syllables of the sentence; only the activation changes.

The reason for having different output units in the two training phases is that Elman relies on the output independence that presumably exists in networks trained with the backpropagation algorithm: because the pre-training phase is performed on an output unit that is different from the output unit used during subsequent training and testing, the pre-training does not change the connection weights between the hidden layer and the second output unit. Therefore, with respect to the second phase of training, all input stimuli are theoretically novel. However, even though the output connection weights are not directly affected by the pre-training phase, the network still embeds knowledge of all input stimuli in its hidden layer activations and connection weights between the input and hidden layers. This means that the pre-training phase indirectly affects the activation of the second output unit.

It is noteworthy that in the case of Elman's network, all input nodes are trained (because of the distributed representation). Therefore, the first interpretation of Marcus' definition of generalization outside the training space does not apply (since there are no entirely untrained nodes). As mentioned in section 3.5 below (Marcus' Evaluation), Marcus does not even question the ability of this network to generalize outside the training space, because he claims that Elman's model is not an example of an eliminative connectionist network. Instead, he argues that it is an implementational model.

3.4 Testing and Results

For testing, Elman generates four sentences, two of them having the form ABA, and the other two ABB. The syllables used in these test sentences are novel with respect to the second training phase, but they are not completely novel to the network. They are part of the initial 120-syllable corpus presented during pre-training.

Table 1 shows Elman's results. He apparently demonstrates that his model is indeed able to do the same kind of discrimination as infants. At a closer look, however, this model's performance is not as good as Elman claims.

Table 1 Elman's results (Elman, 1999; Seidenberg & Elman, 1999)

Test stimuli		Network response	Target
ABA	ba po ba	0.004	0
	ko ga ko	0.008	0
ABB	ba po po	0.853	1
	ko ga ga	0.622	1

3.5 Marcus' Evaluation

In his evaluation of this model, Marcus does not question the actual results reported by Elman (Elman, 1999; Seidenberg & Elman, 1999). Instead, Marcus argues that the reason for Elman's "success" is the fact that the model implements a symbolic mechanism.

As mentioned earlier, Elman employs a preliminary training phase in order to simulate the experience that infants have prior to their participation in the Marcus *et al.*'s (1999) experiment. During this preliminary phase, Elman trains the network to distinguish whether or not two consecutive input stimuli are identical. To Marcus, what

Elman does is equivalent to having an external supervisor that teaches the network to apply a rule of the form "for all [adjacent] syllables **x**, **y**, if **x** = **y**, then output 1 else output 0" (Marcus, 2001). Since the existence of the external supervisor that induces this relation between variables seems to be the reason for the apparent success of the model, Marcus argues that Elman's network is not an example of eliminative connectionism, but a purely implementational connectionist model. According to Marcus, the supervisor of Elman's model may very well make use of abstract relations between variables (i.e., algebraic rules), which are purely symbolic, and these make the entire system, including the supervisor, a symbol manipulator, i.e., the system *implements* a symbolic model by incorporating the capacity to instantiate variables with instances and to manipulate the instances of those variables.

In this argument, Marcus uses the notions "variables" and "symbols". It is certainly possible to define the term "symbol" or "variable" in such a way that all connectionist models make use of them. For instance, one of the definitions Marcus uses for symbols (Marcus, 2001) is that they are context-independent representations of a category (e.g., CATS, DOGS, etc) or individuals (e.g., Felix, Fido, etc). Based on this definition, according to Marcus, any neural network that employs the same set of input units to represent a certain item, regardless of context, makes use of symbols (for example, the word *cat* is always represented by the same set of input units, regardless of its position within a sentence). Taking this argument even further, Marcus considers that in the case of Elman's simple recurrent network, all input items represent instances of the same variable, called *current-word* (or *current-syllable*).

Although I find Marcus' definition of the term "variable" to be tendentious, as Marcus argues, the heart of the matter is not whether a system is presented with context-independent representations that could be viewed as symbols or variables, but whether that system effectively implements operations (e.g., variable binding) over those representations. In the case of Elman's network, it is clear that the existence of the external supervisor that may implement an algebraic rule (as Marcus argues, Elman does not prove that the external supervisor does not implement an operation over variables) may entail that the entire model implements that rule. I will go even further, suggesting that, during the second phase of training, the external supervisor also implements a rule of the form: given three consecutive syllables ("variables") **x**, **y**, and **z**, if $\mathbf{x} = \mathbf{z} \neq \mathbf{y}$, then output 0, else if $\mathbf{x} \neq \mathbf{y} = \mathbf{z}$, then output 1. These rules do exist in the system, and they appear to be enforced by the external supervisor, rather than being implicitly learned by the network. These *explicit* rules seem to be necessary for the success of Elman's network.

3.6 Personal Investigation

Firstly, I would like to emphasize that Elman's simulation differs from Marcus *et al.*'s experiment (1999) in a few major aspects:

- The pre-training phase. It is hard to believe that the infants in the Marcus *et al.*'s (1999) study never heard the individual test syllables before participating in the experiment. Therefore, I think Elman is right in attempting to approximate this prior experience in his experiment. However, the way Elman did it (by training the network to distinguish whether or not two consecutive syllables are identical) is questionable. If 7-

month-old infants know most of the words used in the Marcus *et al.*'s experiment, it is highly unlikely that this knowledge is acquired in terms of similarity between consecutive syllables. At the same time, Elman has not proved that the so-called external supervisor can be implicitly implemented in eliminative networks.

- The network is trained and tested on *both* grammars at the same time. Marcus *et al.*'s experiment is different: the infants are habituated with sentences constructed with only one of the two grammars, and afterwards they are presented with novel sequences of syllables constructed with both grammars. It is arguable whether the infants' task is any easier than the task performed by Elman's network, but, in either case, this important difference makes the outcome of Elman's simulation scarcely significant with regard to the infants' reported behaviour[14].

Secondly, even in the context of Elman's experiment, after further investigation, the reported results proved to be weak. In recent work (Vilcu & Hadley, 2001), we investigated Elman's study, and discovered that, *in general*, his model does not perform as well as Elman (Elman, 1999; Seidenberg & Elman, 1999) claims.

3.6.1 Simulation of Marcus *et al.*'s (1999) experiments

The performance of SRNs (and, in general, any network using the backpropagation algorithm) is influenced by several training parameters, such as initial weights, learning rate, momentum, etc. Usually, the initialization of weights is performed randomly and if training parameters are not chosen properly (especially the learning rate), the network may end up in a region of local minimum of the error function. One way to

[14] However, theoretically, if we ignored the other flaws of Elman's model, his experiment would refute Marcus *et al.*'s theoretical arguments that no eliminative connectionist model is able to differentiate between novel grammatical structures (i.e., generalize outside the training space).

reduce this liability is to perform a batch experiment, i.e., test the network with a large number of different weight initializations and training parameters. Another advantage of this approach is that following the batch sessions we have a more precise and statistically significant picture of the behaviour of the networks, and we also learn whether or not the results are generated fortuitously.

In (Vilcu & Hadley, 2001), we conducted three different batch sessions on Elman's model, using at least 64 networks in each batch. In the first batch session, we used the same training and test stimuli that were employed by Elman (Elman, 1999; Seidenberg & Elman, 1999). During this first simulation, we noticed that even though there is a high degree of overlap among the training patterns (out of 12 bits, only 3 to 4 bits are different), the average distance[15] between the test and training patterns is much higher (6 to 7 bits are different). Since, in our view, the results of this simulation failed to prove Elman's strong claims (see Table 2 for the results of this simulation), in the second batch session we generated different training and test corpora, in an attempt to make those corpora more uniform. Therefore, we manually crafted all training and test stimuli such that the average distance among training patterns (4 to 6 bits) is closer to the average distance between the training and test patterns (6 to 7 bits). Finally, in the third batch session, we generated completely non-overlapping training and test patterns (using localist input representation).

[15] The distance between two vectors is given by the number of vector elements (i.e., bits) by which the vectors differ.

Table 2 **Percentage of the 64 trained networks that have good results during Elman's simulation, according to our evaluation criterion (Vilcu & Hadley, 2001)**

Batch session	Percentage of the 64 trained networks that have good results (%)
1 (training and test corpora are similar to Elman's, 1999)	36
2 (input patterns are uniformly distributed between training and testing, to minimize the differences between the two corpora)	66
3 (no overlap among input patterns)	11

Each network in all three-batch sessions had a different weight initialization, and we generated at least 64 separate trained networks. Otherwise, our experiments were conducted in the same manner as Elman's, i.e., we used the same network structure (12 input units, 10 hidden/context units, and 2 output units), and we followed the same training procedure (starting with pre-training using a 50,000-syllable corpus, then training using thirty-two 3-syllable sentences equally generated with ABA and ABB grammars, and finishing with the test phase involving 2 ABA, and 2 ABB novel sentences). We evaluated the networks' performance with a very lenient criterion, namely that all test sentences must be recognized with at least 50% accuracy[16]. The percentage of trained networks that yield good results according to this criterion is shown in Table 2. We simulated the model with both Elman's TLearn neural network simulator, and my own

[16] Since the network's target output is 0 for ABA sentences and 1 for ABB sentences, we considered that the network produced a good result when both ABA test sentences generated a network output between 0 and 0.5, and both ABB test sentences produced an output between 0.5 and 1. We also used other, less-lenient criterion (e.g., three of the four test sentences are recognized with 65% accuracy, and only one is recognized with 50% accuracy), and then the results were much weaker (less than 10% of the trained networks generated good results).

network simulator (VNNS). There were no statistically significant differences between the results produced by each simulator.

Our results show that Elman's initial claims that his model demonstrates "the ability to generalize a pattern beyond the specific stimuli which gave rise to that generalization" (Elman, 1999) are somewhat overstated. Granted, there is a certain tendency for networks to respond meaningfully, rather than randomly, to the input stimuli, but Elman's reported result is isolated. The extensive experiments reported in (Vilcu & Hadley, 2001) cast serious doubt upon Elman's arguments and the robustness of his result. The aspect regarding robustness is very important because there is a general consensus that learning of language in humans is a robust process. Therefore, all cognitive architectures that simulate language learning need to demonstrate similar levels of reliability.

3.6.2 Simulation of Gomez & Gerken's (1999) experiments

Establishing the limited capabilities of Elman's model with regard to replicating Marcus *et al.*'s (1999) experiment, the next step would be to discover in which degree this model is able to differentiate between more complex grammars, such as the ones employed in the Gomez & Gerken's (1999) experiment. Using exactly the training and test stimuli employed by Gomez & Gerken, I repeated their fourth experiment[17] on a group of 32 Elman networks. None of the 32 networks correctly discriminates between grammatical and ungrammatical test stimuli. A good result is considered whenever the activation value of the second output unit (the one that is trained to output 0 for

[17] The fourth experiment involves two different grammars (Figures 5 and 6). Test sentences use a novel vocabulary.

grammatical sentences and 1 for ungrammatical ones) is less than 0.5 at the end of each of the 10 test grammatical sentences, and greater than 0.5 for each of the 10 ungrammatical test sentences. Based on these results, it is clear that the model fails to learn more complex grammars[18].

Following these latest sets of experiments, the conclusion is that Elman's model not only lacks robustness and generalization abilities, but also its design is somewhat unconvincing. Elman's pre-training procedure is implausible, because it is unlikely that infants acquire their knowledge based on similarities between consecutive words. In addition, Elman's training process is substantially different than Marcus *et al.*'s (1999), because he trains the model on both grammars at the same time. I was not able to significantly replicate his reported result, and the network was unable to differentiate between both complex and simple grammars.

3.7 Knowledge Representation Analysis

In order to investigate the causes for this particular behaviour of Elman's network, I performed several knowledge representation analyses involving hidden activation vectors, and network contributions (products between hidden activations and connection weights). The data has been subjected to both Principal Component Analysis (PCA) and hierarchical clustering. Unless specified otherwise, for each network that is part of these analyses, the hidden activations were collected at the end of each training or test sentence, in order to capture the network's response to the entire sentence.

[18] The simulations of this experiment were performed with my VNNS simulator.

3.7.1 Principal component analysis (PCA)

For one of Elman's networks trained with both ABA and ABB patterns in order to simulate Marcus *et al.*'s (1999) study, performing PCA on the hidden activation vectors generated by the training patterns reveals four principal components. They account for about 95% of the variation existing in the data set. Figure 8 displays the projections of hidden activations onto the first two principal components, along with the input words (syllables) that generate those activations (i.e., the words that appear on the third position of each training sentence). Along the first principal component (which accounts for about 44% of the variation), there is a clear demarcation between the internal representations formed by the words that end in "e" and the words that end in "i". Therefore, the first principal component reflects the separation between the two (and only) vowels that appear in the training set, which means that the network's primary task is to separate the input words based on their vowels. The second principal component (which accounts for about 25% of the variation) reflects the difference in the consonant values existing in the training set (see Figure 8). The demarcation is not as evident as in the case of the first principal component. However, when projected on the second principal component, the hidden activations tend to form several groups, depending on the type of consonants ("d", "j", "l", and "w") that exist in the last words of each training sentence. The words that start with "d" tend to form similar representations as the words that start with "j" and "l", reflecting the fact that the input representations of the "d" and "j" letters, and "d" and "l", respectively, differ in only one bit. On the other hand, *with regard to the second principal component*, the representations of the words that start with "w" are relatively separate from the representations of the words that start with "d", regardless of the vowels that

appear in those words, reflecting the fact that there are many differences between the input representations of those two consonants.

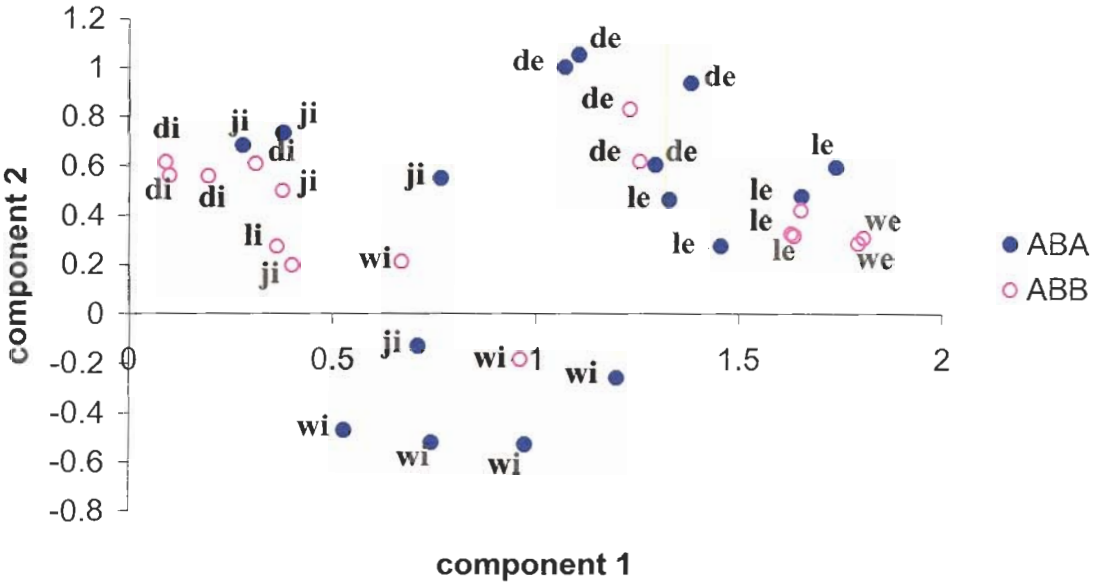

Figure 8 **Projections of hidden activations onto the first two principal components, at the end of each training sentence (the last syllables for both ABA and ABB sentences are shown). The hidden activations separate based on the vowels that appear in the last syllables of each training sentence (along the first principal component), and based on the consonants that appear in those syllables (along the second principal component).**

When projecting the hidden activations onto the third and fourth principal components (which account for about 15% and 11% of the variation, respectively), two slightly separated clusters are formed: the hidden activations generated by ABA words are somewhat separated from the hidden activations generated by ABB words (see Figure 9). However, the two clusters are not very well formed, and are not completely distinctive (the internal representations of about 4 of the 16 input patterns in each condition overlap),

reflecting the fact that the network does not thoroughly succeed in separating the two categories of training sentences.

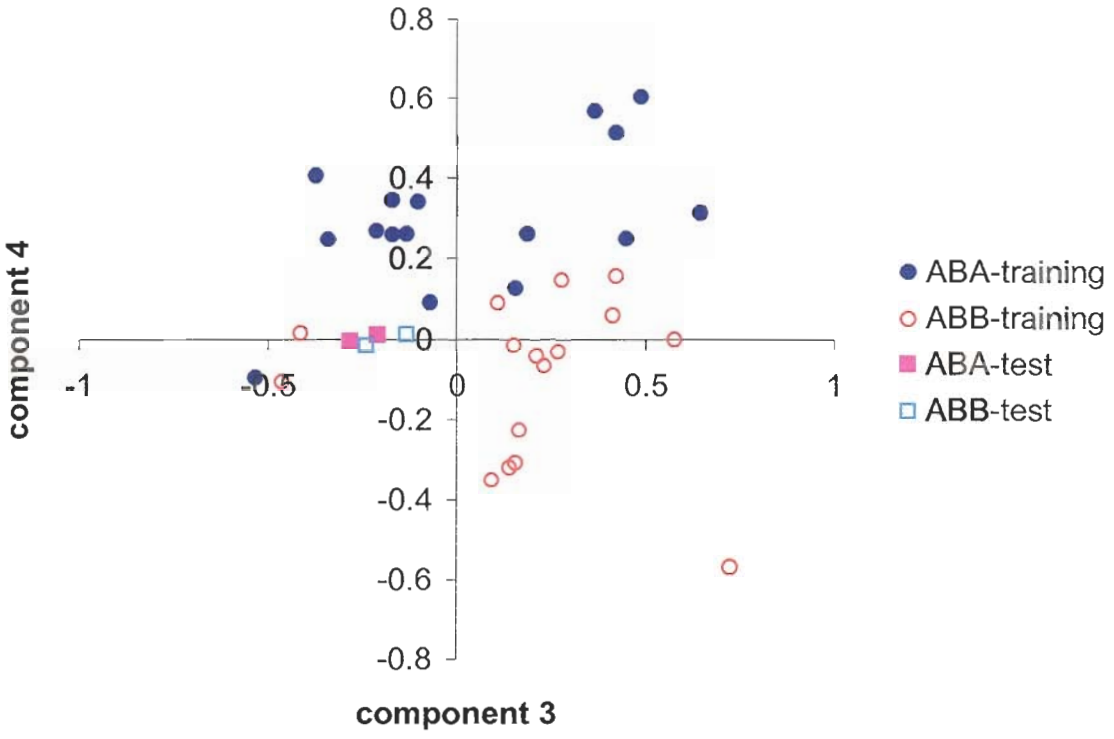

Figure 9 **Projections of hidden activations onto the third and fourth principal components at the end of each training and test sentence (both ABA and ABB sentences are shown). Approximately 25% of the internal representations formed by the ABA training patterns overlap with those formed by the ABB training patterns. The internal representations of the ABA and ABB test patterns are very close to each other, and this makes their separation very difficult.**

In addition to the fact that the network is not able to completely separate the internal representations formed by the ABA training sentences from those formed by the ABB sentences, according to Figure 9, the network has problems separating the test representations as well. Indeed, the internal representations formed by all four-test

sentences are in-between the training clusters, and, also, are very close to each other. Therefore, it is very difficult for the network to categorize the test patterns based on the proximity of their internal representations to the existing (training) clusters. This may explain why many networks fail in distinguishing the two types of grammatical sentences. Also, the fact that only 26% of the network resources (the third and fourth principal components) are allocated to separating the two types of input patterns indicates that this is not the network's main task. According to PCA of hidden activations, the network is mostly concerned with separating the various consonants and vowels which are present in the training set (according to Figure 8, 70% of the network resources are allocated to identifying the training consonants and vowels). In order to robustly generalize to novel input, the network's main objective should be the separation of the two types of grammatical structures. This could explain this model's inability to consistently distinguish between the two grammars, as shown by my numerous experimental results on this model[19].

A similar conclusion can be drawn from the analysis of network contributions (based on the product between hidden activations and output weights). PCA of these contributions reveals two principal components. The projection of contributions onto the first and second principal components contains two somewhat separated clusters (see Figure 10). One cluster is formed by the network contributions produced by ABA patterns, whereas the other one is formed by the network contributions produced by ABB patterns. However, the two clusters do intersect (about 2 of the 16 contributions in each

[19] It is noteworthy that more training does not improve this model's ability to robustly differentiate between the two types of grammatical patterns. I experimented with various numbers of training iterations, and discovered that the performance of this model does not improve when increasing the number of training iterations.

training condition intersect), which means that the network succeeds only partially in separating the two categories of *training* sentences (ABA vs. ABB). During testing, the network may classify the test stimuli to either one of the two grammars, depending on which cluster the test representations are closer to. As shown in Figure 10, the network contributions produced by Elman's *test* stimuli are located *exactly* at the intersection between the *two training clusters*, making the separation of *test* stimuli very difficult and somewhat arbitrary.

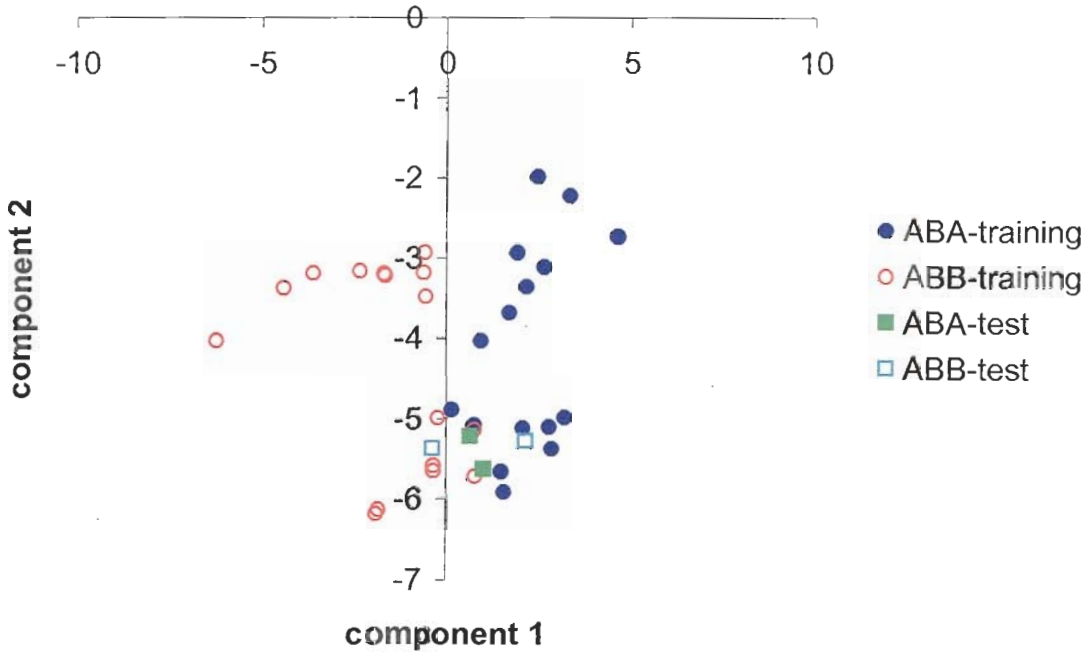

Figure 10 **Projections of network contributions onto the first two principal components during training and testing (both ABA vs. ABB patterns are shown). About 12.5% of the network contributions formed by the ABA training sentences overlap with those formed by the ABA training sentences. The network contributions formed by the test patterns in each condition are very difficult to separate.**

Based on the analysis performed up to this point, we would expect the network's performance to deteriorate even more when using more complex grammars (e.g., Gomez & Gerken's grammars). If in the simple case (ABA vs. ABB) there is a certain tendency of the network to form some meaningful internal representations during training (according to Figures 8, 9, 10), this tendency is completely lost when dealing with more complex grammars (see Figures 11 and 12). The internal representations formed during training with Gomez & Gerken grammars (1999) do not have any coherence with regard to the first four principal components produced by the PCA of hidden activations (the four principal components account for about 95% of the variation).

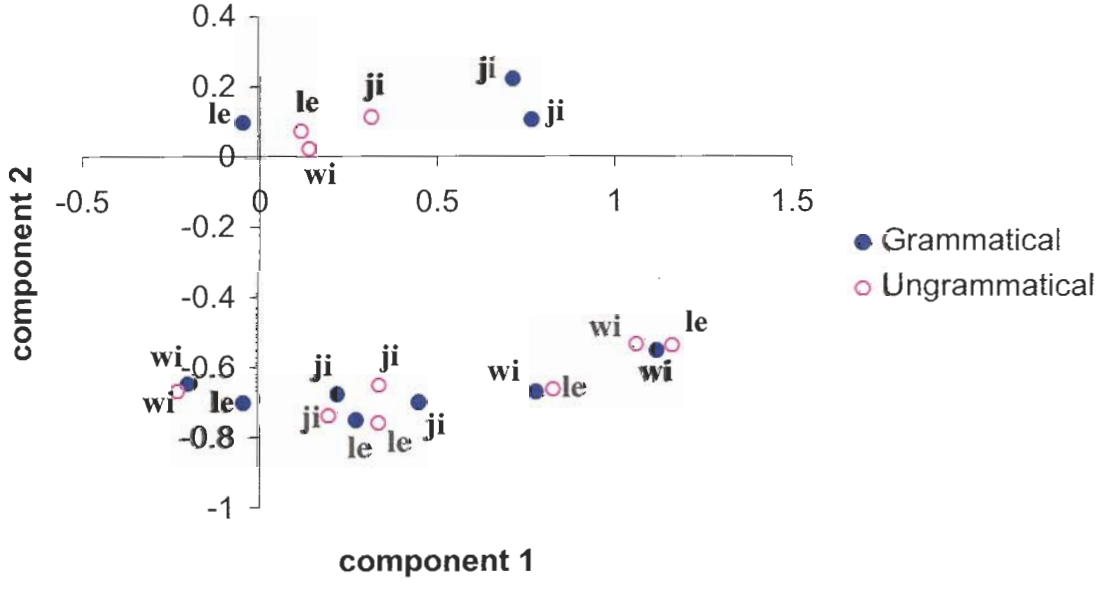

Figure 11 **Projections of hidden activation onto the first two principal components (Gomez & Gerken grammars). There is no meaningful separation among the internal representations formed by grammatical and ungrammatical training patterns.**

Figures 11 and 12 display the projections of internal representations onto the first four principal components *at the end of each training sentence* formed by Gomez & Gerken grammars. Figure 11 also shows the input words that generate those representations (i.e., the words that appear on the last position of each sentence). There is no meaningful separation among the internal representations of various training patterns with regard to vowels, consonants, groups of words, etc., or between patterns constructed with one grammar or the other. The internal representations formed by the network at the end of each training sentence are quite unreliable, and this may explain why the network fails to differentiate between more complex grammars.

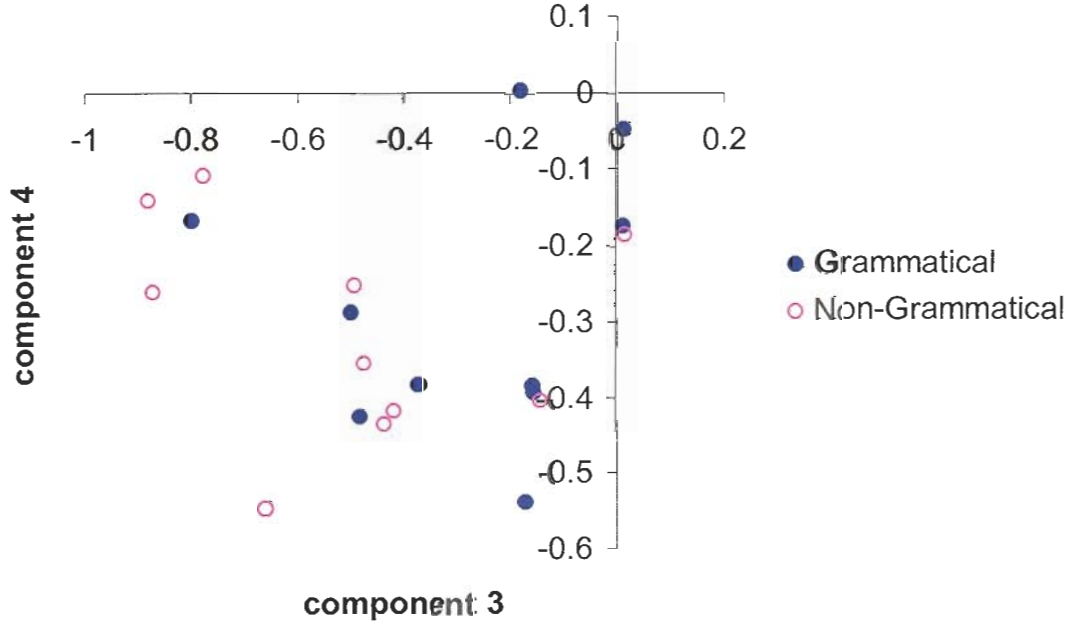

Figure 12 **Projections of contributions onto the third and fourth principal components (Gomez & Gerken grammars). There is a high degree of overlap between the network contributions formed by grammatical patterns and those formed by ungrammatical patterns.**

3.7.2 Hierarchical clustering

In order to show how Elman's model groups the test patterns, a cluster analysis has been performed on the hidden activations generated at the end of each test sentence on 8 separate networks trained with both ABA and ABB patterns. Because, originally, there are only 2 ABA and 2 ABB test sentences, to increase the statistical significance of this analysis, 6 new test sentences have been generated in each condition. The new test sentences were formed with words that did not occur in the training set (but they were part of the preliminary training corpus). Figures 13-20 display the results of the cluster analysis performed on 16 hidden activation vectors, generated at the end of 8 ABA and 8 ABB test sentences, for 8 different networks. Since I am mostly interested in finding whether the networks can group the hidden activations into two clusters (ABA vs. ABB), I forced each cluster analysis to create two groups. In two different colours, Figures 13-20 display how the networks associate each hidden activation vector to one of the two clusters. As shown in Figures 13-20, none of the 8 networks is able to correctly separate all 16-test stimuli into two groups. There is an important level of overlap (see Figures 13-20 for details) between the hidden activations formed by ABA and ABB test patterns, and this confirms the previous argument that the model is unable to robustly separate the test stimuli.

The following 8 graphs (Figures 13 to 20) display the results of cluster analysis for 8 different networks (denoted network #1 to network #8).

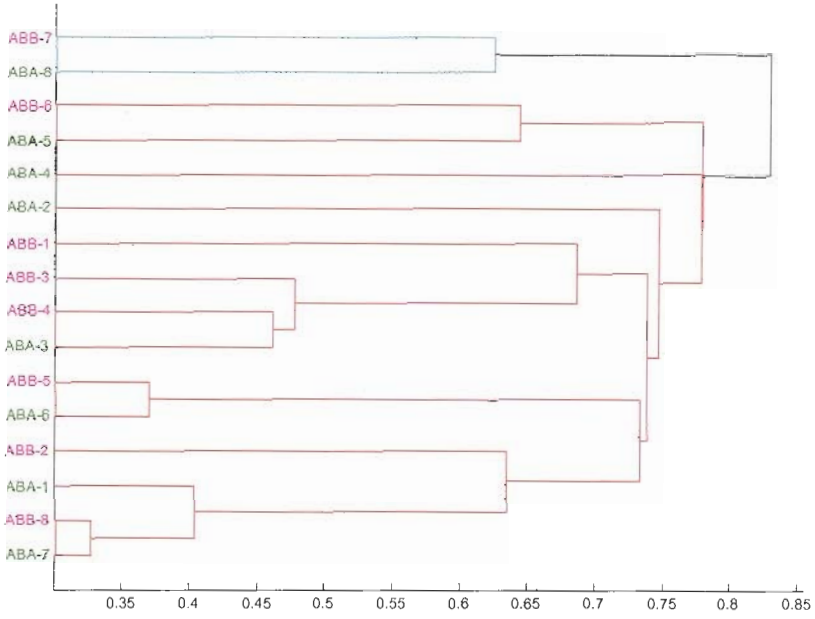

Figure 13 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #1 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there is one ABA and one ABB sentence (50% ABA), whereas in cluster 2 (red) there are 7 ABA and 7 ABB sentences (50% ABA).**

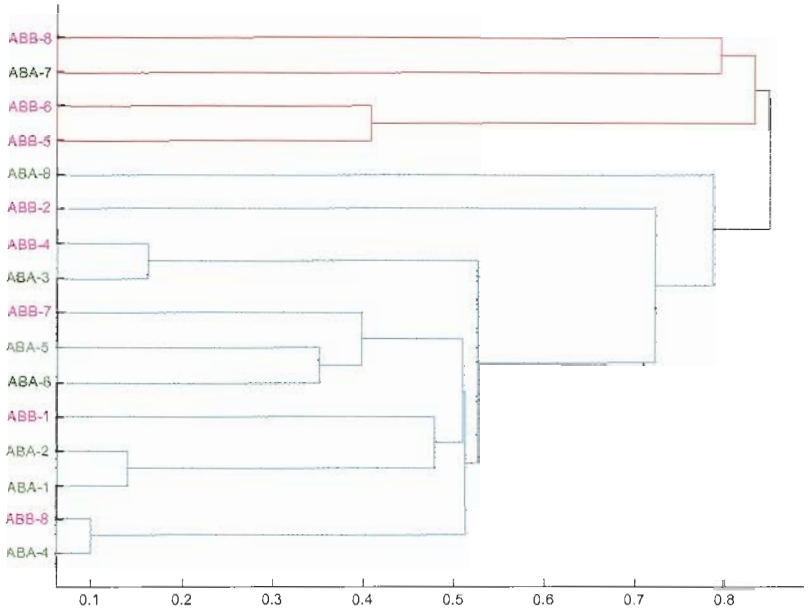

Figure 14 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #2 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there are 5 ABA and 7 ABB sentences (42% ABA), whereas in cluster 2 (red) there are 3 ABA and 1 ABB sentence (75% ABA).**

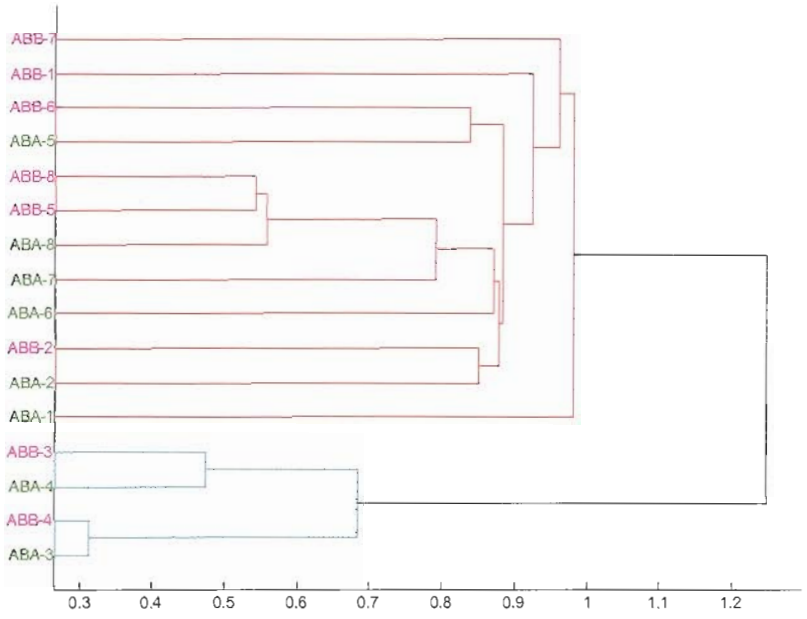

Figure 15 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #3 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there are 2 ABA and 2 ABB sentences (50% ABA), whereas in cluster 2 (red) there are 6 ABA and 6 ABB sentences (50% ABA).**

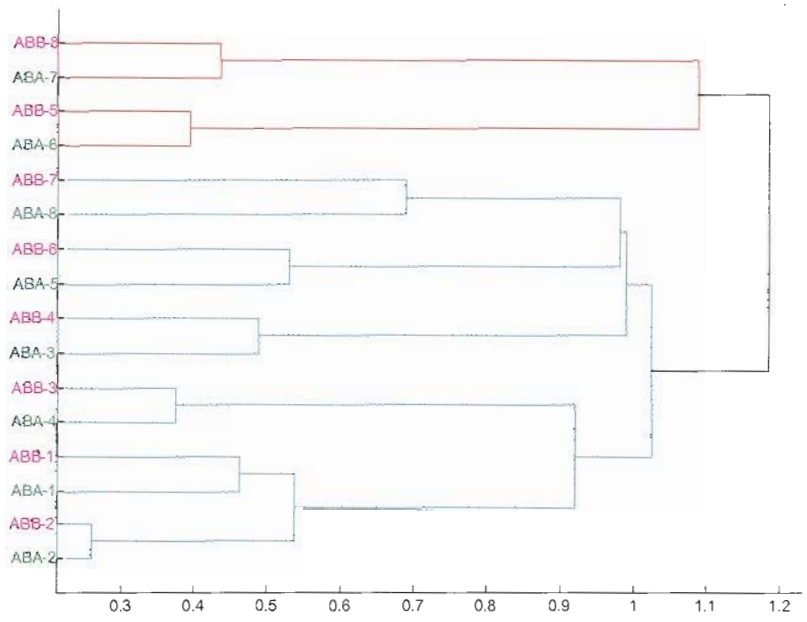

Figure 16 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #4 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there are 6 ABA and 6 ABB sentences (50% ABA), whereas in cluster 2 (red) there are 2 ABA and 2 ABB sentences (50% ABA).**

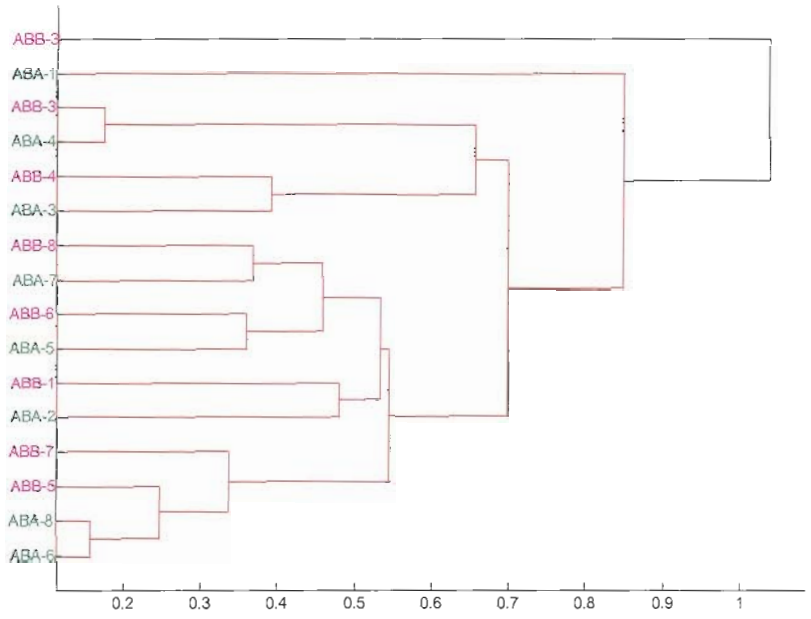

Figure 17 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #5 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 there is only 1 ABB sentence (100% ABB), whereas in cluster 2 (red) there are 8 ABA and 7 ABB sentences (47% ABB).**

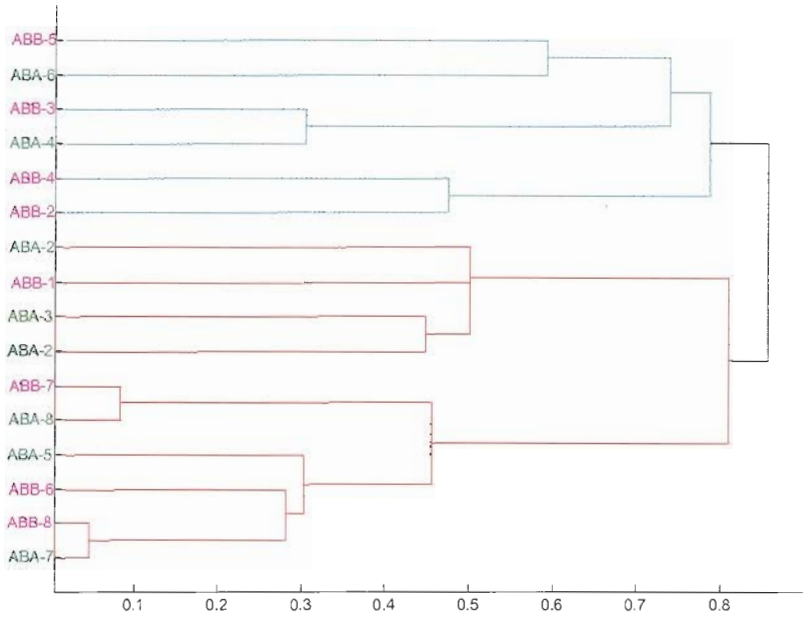

Figure 18 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #6 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there are 3 ABA and 2 ABB sentences (60% ABA), whereas in cluster 2 (red) there are 5 ABA and 6 ABB sentences (45% ABA).**

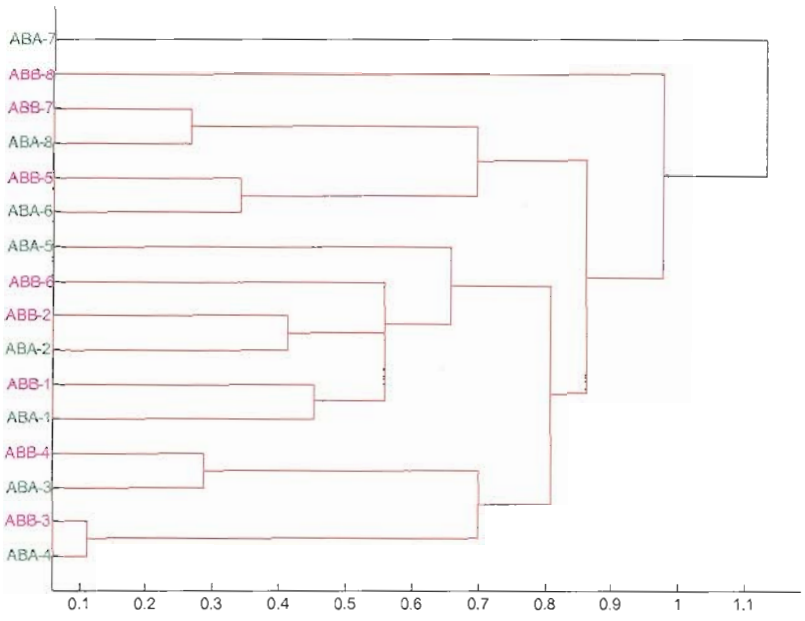

Figure 19 **The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #7 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 there is only 1 ABA sentence (100% ABA), whereas in cluster 2 (red) there are 7 ABA and 8 ABB sentences (47% ABA).**

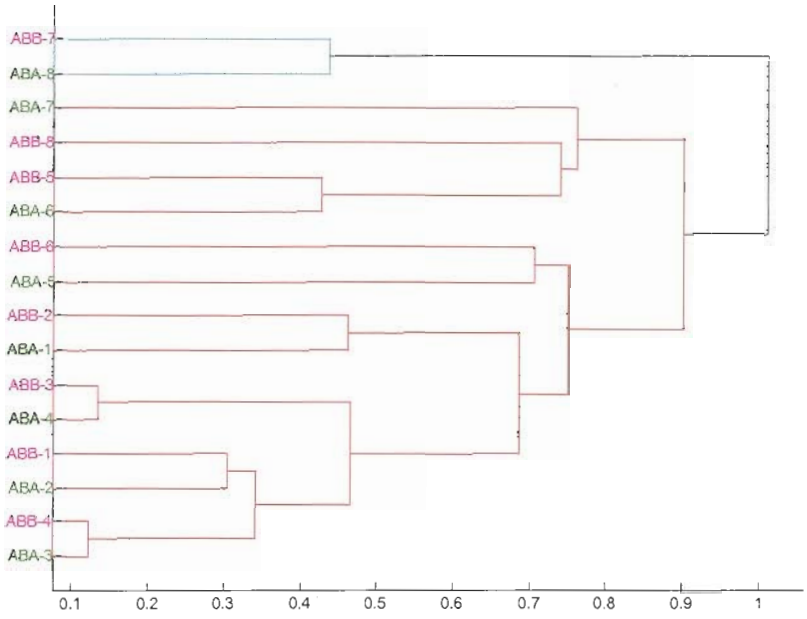

Figure 20 The results of cluster analysis performed on 16 hidden activation vectors at the end of 8 ABA and 8 ABB test sentences for network #8 (the horizontal values represent the Euclidian distances between vectors). In cluster 1 (blue) there is 1 ABA and 1 ABB sentence (50% ABA), whereas in cluster 2 (red) there are 7 ABA and 7 ABB sentences (50% ABA).

In order to see how this connectionist model groups the internal representations during testing with Gomez & Gerken's (1999) grammars, a cluster analysis has been performed on 20 hidden activation vectors formed at the end of 10 grammatical and 10 ungrammatical test sentences. Since I am mostly interested in finding whether the network is able to cluster the hidden activations into two groups (grammatical vs. ungrammatical), Figure 21 displays, in two different colours, the results of the cluster analysis, along with information regarding how the network generates the two clusters and how it assigns each hidden activation vector to one of the two clusters. As seen in Figure 21, there is a very high level of overlap between hidden activations formed by

grammatical and ungrammatical test sentences, confirming that the network has difficulties distinguishing between the two categories of sentences.

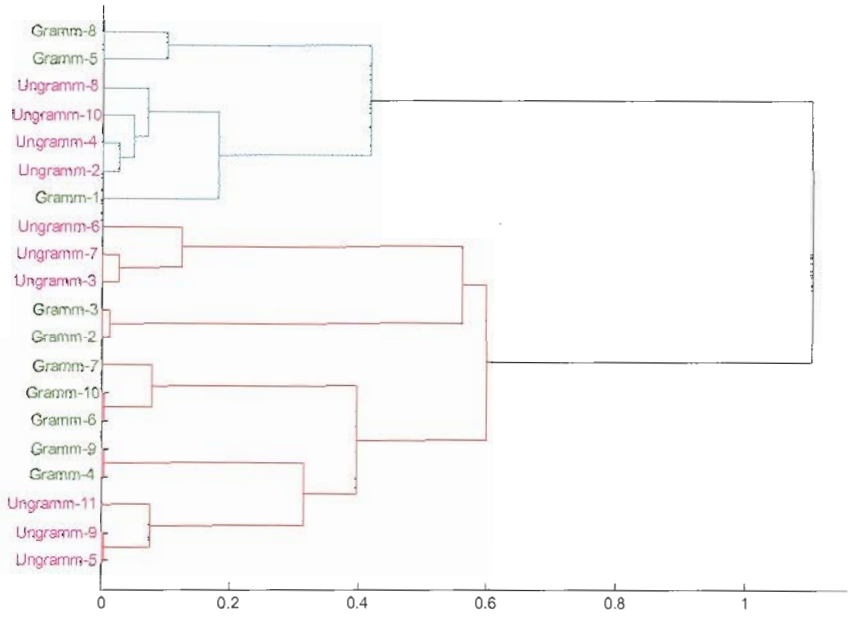

Figure 21 The results of binary cluster analysis on 20 hidden activation vectors formed at the end of 10 grammatical (Gramm-1 to Gramm-10) and 10 ungrammatical (Ungramm-1 to Ungramm-10) test sentences for one network trained on Gomez & Gerken's grammars (the values on the horizontal line represent the Euclidian distances between vectors). In cluster 1 (blue) there are 3 grammatical and 4 ungrammatical sentences (43% grammatical), whereas in cluster 2 (red) there are 7 grammatical and 6 ungrammatical sentences (54% grammatical).

According to Figure 21, the way the network groups the internal representations is rather arbitrary. For instance, in the case of cluster 1 (blue), the internal representation formed by the second ungrammatical test sentence (Ungramm-2) is very close to the internal representations of 3 other ungrammatical sentences (Ungramm-4, Ungramm-10, and Ungramm-8), but also groups with those formed by grammatical sentences Gramm-1, Gramm-5, and Gramm-8. In cluster 2 (red) there is also a great level of overlap. The

subgroup formed by Gramm-4 and Gramm-9 firstly clusters with the subgroup formed by ungrammatical sentences Ungramm-5, Ungramm-9, and Ungramm-11 before clustering with Gramm-6, Gramm-10, and Gramm-7. Also, the grammatical subgroup Gramm-2 and Gramm-3 clusters with the ungrammatical subgroup Ungramm-3, Ungramm-7, and Ungramm-6.

4 SHULTZ', AND SHULTZ & BALE'S MODELS

Shultz' (1999), and Shultz & Bale's (2001) simulations employ an encoder version of the cascade-correlation learning algorithm. Shultz & Bale (2001) use the same model as in Shultz (1999), the only difference being the input representation.

4.1 Network Architecture and Preliminaries

The cascade-correlation (Fahlman & Lebiere, 1990) is a generative algorithm in feed-forward networks. The initial network structure only contains the input and output layers, without any hidden units. New hidden units are added to the network during training, each hidden unit on a separate layer. During training, the learning algorithm uses a set of hidden units (called candidate units), which are not part of the network structure yet. At each time step, each candidate unit is temporarily added to the network, one at a time, and the correlation between the activation of the newly added candidate unit and the network error is computed. The candidate unit that has the highest correlation (i.e., the unit whose activation minimizes the network error the most among all other candidate units) is permanently added to the network. Each new unit resides on its own hidden layer, and is fully connected to all the input and all the existing hidden units (see Figure 22). The network is trained to output the same pattern that is presented at input.

Shultz & Bale (2001) maintain that this learning algorithm is "neurologically plausible"[20], and may account for some cognitive processes, such as neurogenesis and synaptogenesis. T. R. Shultz and others have extensively used this algorithm in recent years, to simulate various aspects of cognitive development in children. In the present cases (Shultz, 1999; Shultz & Bale, 2001), they employ an "encoder" version of the cascade-correlation model. This version precludes direct input-output connections, in order to avoid generating networks simply having connections of weight 1 between the input and output layers (see Figure 22).

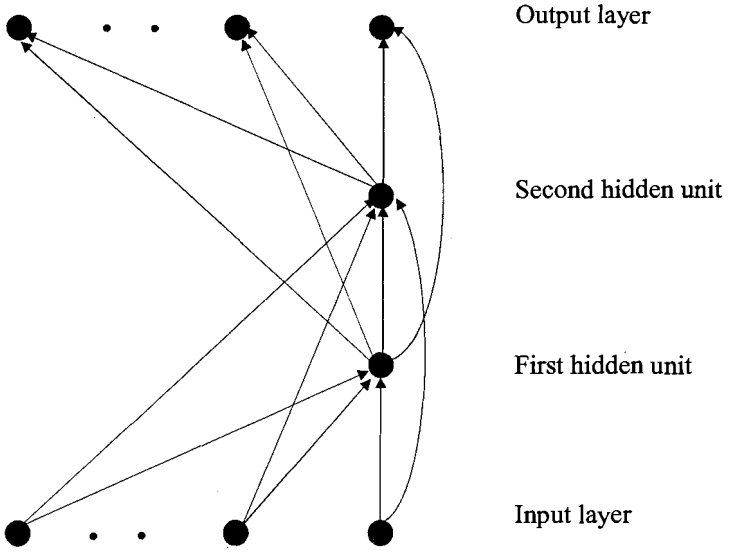

Figure 22 The cascade architecture, after adding 2 hidden units (not all the units from the input and output layers are shown)

[20] However, the error corrections and error measurement that the algorithm uses are hard to justify biologically.

4.2 Input Representation

The novelty of Shultz & Bale's simulation (2001) over Shultz' (1999) lies in the input representation. In the newer study, they use a sonority scale to encode each phoneme of the data set. The choice of this encoding reflects the fact that sonority represents the "quality of vowel likeness" (Shultz & Bale, 2001), i.e. some phonemes can be considered to be "more vowel-like" than others. The sonority scale ranges from "low vowels", such as /a/ and /æ/ that were assigned a sonority of +6.0, to "voiceless stops", such as /p/, /t/, and /k/ that were assigned a sonority of –6.0. For example, a few of the 2-letter words (syllables) used in the simulation and their encoding are: *ga* = -5.0 6.0, *wo* = -1.0 5.0, *ti* = -6.0 4.0.

A sentence consists of three such syllables, generated using one simple grammar (having the form ABA, ABB, or AAB). E.g.: *ga ti ga* = -5.0 6.0 -6.0 4.0 -5.0 6.0, *li na na* = -1.0 4.0 -2.0 6.0 -2.0 6.0.

In his earlier work, Shultz (1999) assigns an odd number between 1 and 7 to category "A" syllables, and an even number between 2 and 8 to category "B" syllables. For example: *ga ti ga* = 1 2 1, *li na li* = 3 4 3, *ga ti ti* = 1 2 2, *ga na na* = 1 4 4. The syllables used during testing are represented by values interpolated within the training patterns. For example: *wo fe wo* = 2.5 3.5 2.5, *de ko ko* = 5.5 6.5 6.5.

4.3 Training, Testing, and Results

Similar to Marcus *et al.* (1999), Shultz & Bales' (2001) simulation also consisted of three experiments[21], each experiment involving 16 separate networks (one network

[21] In Shultz (1999), the second experiment of Marcus *et al.* (1999) is not performed.

corresponds to one infant). The first two parts (experiments 1 and 2) involved training eight networks with 16 sentences generated by the ABA grammar, and another eight networks trained with 16 ABB sentences. All 16 networks were then tested with four novel sentences derived from both grammars (2 sentences with each grammar). Part three (experiment 3) was similar to the first two, except that the grammars involved were AAB and ABB.

In contrast to Elman (Elman, 1999; Seidenberg & Elman, 1999), Shultz (1999) and Shultz & Bale (2001) do not pre-train their model. They start directly with the training of networks using exactly the stimuli employed in Marcus *et al.*'s (1999) experiments. The training sentences are presented one at a time, the networks having three input and three output units (one unit per syllable, in the case of the Shultz's model, 1999), or six input and six output units (one unit for each phoneme in the sentence, in the case of the Shultz & Bale's model, 2001). Shultz (1999), and Shultz & Bale (2001) use the same training parameters as Fahlman & Lebiere's (1991) default values, except for a score-threshold of 0.8, and input-patience and output-patience of 1. The score-threshold represents the tolerated difference between target and actual outputs, and it was raised from 0.4 in order to reduce the crispness of the knowledge representations. The patience parameter represents the number of epochs allowed to pass with little increase in error reduction or correlation, before shifting from input phase to output phase, or vice-versa[22].

[22] The cascade-correlation algorithm alternates between two phases: output phase and input phase. During the output phase, connection weights entering the output units are adjusted in order to minimize the network error. During the input phase, connection weights entering candidate-hidden units are adjusted in order to increase the correlation between network error and activation of the candidates. Shultz & Bale state that both patience parameters are reduced to 1 because the network performance does not improve much after it fails to improve on a single epoch.

Initially, the networks do not have any hidden units, these units being added during training, according to the cascade-correlation algorithm (i.e., hidden units are added in order to minimize the network error at each time step). The networks are trained to output the same pattern that is presented at the input layer, and it is shown that, at the end of training, the algorithm optimally generates two hidden units. During testing, Shultz (1999), and Shultz & Bale (2001) measure the network error when presented with novel, familiar (having the same structure as the training grammar) and unfamiliar sentences, and report fairly good results. Table 3 displays the average network error, i.e., the average of the square root errors between the actual output and target vectors of each network, during the test phase, reported by Shultz (1999) and Shultz & Bale (2001).

Table 3 Average network error reported by Shultz (1999), and Shultz & Bale (2001)

Experiment	Test vs. training sentences	Network error (Shultz, 1999)	Network error (Shultz & Bale, 2001)
1 ABA vs. ABB	Familiar	0.649	8.2
	Unfamiliar	1.577	14.5
2 ABA vs. ABB	Familiar	not available	13.1
	Unfamiliar	not available	15.8
3 AAB vs. ABB	Familiar	0.570	12.9
	Unfamiliar	1.491	15.3

4.4 Marcus' Evaluation

Apart from the input representation, Shultz' (1999) and Shultz & Bale's (2001) simulations are conducted in the same way as each other, and generate similar results. Marcus' opinion about the new Shultz & Bale's (2001) experiment is not known. However, Marcus did reply to the original Shultz' work (1999), and argued that the main reason for the success of that network is the fact that "Shultz uses each node as a variable

that represents a particular position in the sentence" (Marcus, 2001). In other words, to Marcus, Shultz' model (1999) is not an example of eliminative connectionism, but an illustration of how connectionist models can successfully *implement* classical algorithms, namely copying the contents of one variable to the contents of another. Specifically, Marcus claims that each of the 3 input/output nodes in Shultz' network represents a variable corresponding to the word position in the input sentence[23], and the nodes get instantiated with different values, depending on the sentence being currently presented. Marcus argues that what the model may be doing is to copy the activations of the input layer (the contents of three independent variables) to the output layer, as in Figure 23.

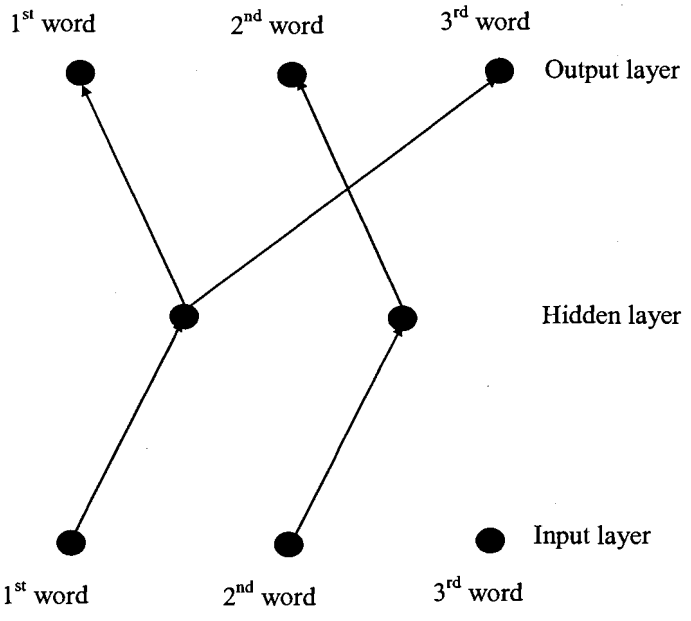

Figure 23 A simplified version of Shultz' model (1999), according to Marcus' conjecture (2001)

[23] The first node represents the variable *first-word*, the second node represents the variable *second-word*, while the third node represents the variable *third-word*.

More specifically:

- The activations of the first two input nodes are copied to two hidden units.
- The third input node develops an insignificant connection weight (close to 0) towards the units from the hidden layer, i.e., it does not participate in subsequent training and testing, and it does not have any influence on the network's results.
- The activation of the first hidden unit is equally copied to the first and third output units, while the activation of the second hidden unit is copied to the second output unit. The connection between the first hidden unit and the second output unit, and the connections between the second hidden unit and the first and third output units are not significant during training and testing.

I agree that Shultz' model (1999) *could* work the way Marcus suggests, and it is possible that the way Shultz designed his network can lead to the conclusion that it manipulates variables. It is noteworthy that there is one major difference between how this model makes use of "variables" and Elman's (1999): if the training space of interest is the space of all sentences, and a "variable" is associated with each word of the sentence, then a triplet of such variables identifies a point in that space. Since the model receives the entire triplet of "variables" all at once, it would be possible to implicitly implement the kind of operation that Marcus suggests. However, as I argue below, I think Marcus may over-simplify this model, and unnecessarily reduce it to a classical implementation. I believe the model does not do variable binding. My personal investigation on this model reveals that the connection weights develop in a much more

complex way that in Marcus' conjecture, and that the model does not do operations over variables.

4.5 Personal Investigation

I would like to emphasize that there are a few factors that render Shultz' (1999), and Shultz & Bale's (2001) studies better than Elman's (Elman, 1999; Seidenberg & Elman, 1999), with regard to replicating Marcus *et al.*'s experiments (1999):

- There is not a preliminary training phase; the network performance is based entirely on input representation and training phases. From this point of view, the simulations performed by Shultz (1999) and Shultz & Bale (2001) follow Marcus *et al.*'s (1999) closer than Elman (Elman, 1999; Seidenberg & Elman, 1999) does.
- They use a generative algorithm (which, arguably, generates the best possible network structure for the current task).
- Shultz & Bale (2001) perform all three experiments of Marcus *et al.*'s (1999), using a similar procedure as in Marcus *et al.*. Elman (Elman, 1999; Seidenberg & Elman, 1999) only performs experiment 2, and, as specified in the previous chapter, trains the network on both grammatical structures at the same time.
- The reported results are closer to the infants' results.

4.5.1 Simulation of Marcus *et al.*'s (1999) experiments

In recent work (Vilcu & Hadley, 2003), we took a closer look at both Shultz' (1999), and Shultz & Bale's (2001) studies. We performed the same experiments that

Shultz (1999), and Shultz & Bale (2001) did, using various input corpora. To simulate their studies, I implemented the original cascade-correlation algorithm (Fahlman & Lebiere, 1990) within VNNS, and made those changes dictated by Shultz (1999) and Shultz & Bale (2001) specific models: eliminated the direct input-output connections, modified the score-threshold to a value of 0.8, and input-patience and output-patience to 1. When we employed the same input corpora as in Shultz (1999), and Shultz & Bale (2001), we were able to replicate their reported results. However, we noticed that the connection weights developed in a more complex way than in Marcus' conjecture (Figure 23). The input nodes' activation is not simply copied from one unit to another, and all the connection weights developed during training are significant, all nodes being fully connected and participating in training and testing[24]. This means that Marcus' suggestion of how this model behaves is mistaken.

According to Marcus, Shultz uses a "one-node-per-variable encoding scheme" (Marcus, 2001), where each node corresponds to one "variable" (first word, second word, and third word), and the instances of those variables are copied to the hidden layer: the instance of variable "first word" is copied to the first hidden unit, whereas the instance of variable "second word" is copied to the second hidden unit. In other words, Marcus claims that there is a simple and straightforward mapping between the "variables" encoded on the input layer, and the "variables" encoded on the hidden layer. However, our investigation shows that that is not the case. First of all, the third input is fully connected to both hidden units, and secondly, the network's layers are connected in a

[24] As an example, the third input node has significantly weighted connections with the hidden units, and participates in all phases of training and testing.

complex way. We believe that there are no apparent mappings between instances of "variables" is Shultz' network.

Table 4 The original and the new test patterns in experiment 1, 2, and 3 when testing interpolation on Shultz & Bale's model (the underlined values are changed)

	Original test patterns		New test patterns	
	Letter representation	Numerical representation	Letter representation	Numerical representation
1	wo fe wo de ko de wo fe fe de ko ko	-1 5 -4 5 -1 5 -5 5 -6 5 -5 5 -1 5 -4 5 -4 5 -5 5 -6 5 -6 5	wo fe wo de ko de vo fe fe de ko ko	-1 5 -4 5 -1 5 -5 5 -6 5 -5 5 -3 5 -4 5 -4 5 -5 5 -6 5 -6 5
2	ba po ba ko ga ko ba po po ko ga ga	-5 6 -6 5 -5 6 -6 5 -5 6 -6 5 -5 6 -6 5 -6 5 -6 5 -5 6 -5 6	ma po ma ko ga ko ba po po ko ga ga	-2 6 -6 5 -2 6 -6 5 -5 6 -6 5 -5 6 -6 5 -6 5 -6 5 -5 6 -5 6
3	ba ba po ko ko ga ba po po ko ga ga	-5 6 -5 6 -6 5 -6 5 -6 5 -5 6 -5 6 -6 5 -6 5 -6 5 -5 6 -5 6	ma ma po ko ko ga ba po po ko ga ga	-2 6 -2 6 -6 5 -6 5 -6 5 -5 6 -5 6 -6 5 -6 5 -6 5 -5 6 -5 6

Although Shultz (1999) and Shultz & Bale (2001) come close to replicating the results reported by Marcus *et al.* (1999), these models have limited generalization capabilities. In Vilcu & Hadley (2003) we altered the training corpus (by changing as little as one test syllable), and discovered that the models fail to recognize the two syntactic structures when presented with certain kinds of novel sentences that were not part of the training examples. All these new simulations were performed on 32 different networks, and the average network error (the average of the root square errors between the actual network outputs and the target ones) was measured. In experiment 1, we initially replaced just one instance of the letter "w" (sonority of -1.0) with "v" (sonority

of -3.0) in one of the ABB test sentences (see Table 4). In experiments 2 and 3, we replaced two instances of the same letter "b" (sonority -5.0) with the letter "m" (sonority -2.0) in one ABA test sentence (experiment 2), and one AAB test sentence (experiment 3). Table 4 shows these changes for all three experiments.

Table 5 Average of network error in the three experiments of Shultz & Bale using altered test patterns (both the interpolation and extrapolation abilities are tested)

Experiment	Test vs. training sentences	Altered test patterns	
		Sonority test values within the range of the sonority scale used in training (i.e., testing interpolation)	Sonority test values outside the range of the sonority scale used in training (i.e., testing extrapolation)
1	Grammatical	8.83	97.29
	Ungrammatical	8.46	89.83
2	Grammatical	14.56	136.89
	Ungrammatical	13.83	122.83
3	Grammatical	14.71	144.82
	Ungrammatical	14.57	129.09

In experiment 1, the novel value -3.0 (letter "v", or "z") represents the sonority average of all the consonants that are presented to the input layer during training. It can be considered a "generic" consonant. If the model had genuinely learned the underlying structure of the input patterns, and if it was trained on ABA patterns, it should have no difficulty in distinguishing between the unchanged ABA test sentences and the novel ABB sentences that contained one new consonant. Since this new letter was novel to the network, one would expect the error to be higher for the novel ABB sentences that contained it, along with even better differentiation between the familiar ABA sentences and the unfamiliar ABB sentences. In reality, our results showed that the error for the unfamiliar ABB patterns was smaller than for the ABA patterns. Although changes to the

test corpus were minimal, and all new values were well within the training space, the model was not able to differentiate the two categories of sentences in any of the three experiments. Table 5 shows the results of our simulations on each of the three experiments using the altered test set.

One may argue that the new ABB sentence (*vo fe fe*: -3.0 5.0 -4.0 5.0 -4.0 5.0) in experiment 1 is difficult to distinguish because the new "A" syllable is close to the "B" syllables. The new "A" syllable may be closer to the "B" syllables, but so are another two of the four test sentences in experiment 1, and all original test sentences in experiments 2 and 3. The network should deal with these small phonetic differences the same way as the Marcus *et al.*'s infants do. Moreover, as described below, the model likewise failed to differentiate between the ABA and ABB sentences even when we made a different change, i.e., we replaced consonant "f" (-4.0) with "v" (-3.0) in the first ABB test sentence: *wo ve ve* (-1.0 5.0 -3.0 5.0 -3.0 5.0) replaced *wo fe fe* (-1.0 5.0 -4.0 5.0 -4.0 5.0).

To address another possible concern (i.e., that the previous experiment may still be unfair because only one ABB sentence has been altered, whereas none of the ABA test sentences were changed), we performed yet another experiment and changed one sentence in each of the two conditions by replacing consonant "f" (-4.0) with "v" (-3.0). The new sentences are *wo ve wo* (-1.0 5.0 -3.0 5.0 -1.0 5.0) and *wo ve wo* (-1.0 5.0 -3.0 5.0 -3.0 5.0), which replace *wo fe wo* and *wo fe fe*, respectively. The new "A" and "B" syllables are not close any longer, and the change affected both categories of sentences. But the model still failed to differentiate between the two categories (the average network error was greater for familiar test sentences: 7.86 vs. 7.36 in the case of the unfamiliar

test sentences). Note that at this point we had tested the network for experiment 1 with just as many novel sentences as Shultz & Bale (2001) had used.

In experiments 2 and 3, one instance of the familiar consonant "b" (-5.0) was replaced with the unfamiliar consonant "m" (-2.0). Thus, instead of having *ba po ba*, we now have *ma po ma* in experiment 2, and *ma ma po*, instead of *ba ba po*, in experiment 3. Admittedly, we cannot say with certainty what infants would have done if they were presented with these new test sentences. However, to an adult the changes seem minimal, and it is entirely credible that infants would still have been able to differentiate between the familiar and unfamiliar sentences. In contrast, the networks exhibit a smaller error for unfamiliar test sentences (14.23 in experiment 2, and 14.96 in experiment 3) than for familiar ones (14.67 in experiment 2, and 15.12 in experiment 3).

A similar result is obtained when additional novel test sentences are employed. For example, replacing consonant "b" with "m" in the first ABB test sentence (-2.0 6.0 -6.0 5.0 -6.0 5.0 instead of -5.0 6.0 -6.0 5.0 -6.0 5.0), and consonant "g" with "f" in the second ABA/AAB test sentence (-6.0 5.0 -4.0 6.0 -6.0 5.0 instead of -6.0 5.0 -5.0 6.0 -6.0 5.0) result in a smaller network error for unfamiliar test sentences than for familiar ones (14.93 vs. 15.47). Contrary to their claim, our results show that Shultz & Bale's model is not able to "generalize [...] to novel sentences within the range of the training patterns" (Shultz & Bale, 2001). Note that these results are statistically significant. We have employed as many novel sentences for experiments 2 and 3 as did Shultz & Bale.

In order to test the extrapolation ability of the model, we picked 6 different values outside the sonority scale: 4 of these values were below -6.0 and were used as consonants, and the other 2 were greater than +6.0 and were used as vowels (see Table

6). These values were used to create 4 test sentences (2 test sentences with the familiar grammar, and 2 test sentences with the unfamiliar grammar) for all three experiments. Note that the number of test sentences is the same as in the Shultz (1999) and Shultz & Bale (2001) cases.

Table 6 Sonority values used when testing extrapolation in Shultz & Bale's (2001) model

Category A		Category B	
Consonant	Vowel	Consonant	Vowel
-10.0	8.0	-9.0	8.0
-8.0	7.0	-7.0	7.0

For instance, an ABA and an ABB sentence would be: -10.0 8.0 -9.0 8.0 -10.0 8.0, and -10.0 8.0 -9.0 8.0 -9.0 8.0.

As shown in Table 5, the average network error for all three experiments was *smaller in the unfamiliar case.* Thus, the network's ability to extrapolate is apparently weak, and the model does not reliably "recognize syntactic differences in sentences containing words with sonorities outside of the training range" (Shultz & Bale, 2001).

One may argue that the sonority values used in this experiment are rather far from the training values, and this could explain why the networks are not able to extrapolate well. However, if the model really had learned the grammar, and the hidden nodes genuinely had acted as "category" nodes for the "A" and "B" words as Shultz & Bale claim, then the network should not have difficulty in dealing with those "extreme" sonority values.

4.5.2 The peaks vs. valleys experiment

One of the most important particularities that we discovered for both Shultz' (1999) and Shultz & Bale's (2001) models is the fact that the networks are driven by the numerical shapes (contours) of the input patterns, rather than by their grammatical structure. For instance, in the case of Shultz' model (1999), we performed a simple experiment by training the networks with ABA patterns that have the following numerical contour: the numerical representation of all syllables that denote the "B" category are greater than the numerical representation of the syllables that symbolize the "A" category (i.e., they form a shape of a "peak"). We then performed two test simulations. In the first simulation, we generated two novel ABA test sentences, and two novel ABB test sentences. The two ABA test sentences had a familiar shape (peaks) (see Table 7). In the second test simulation, we generated two new ABA test sentences having an unfamiliar shape (valleys)[25], and used the same ABB test sentences as in the first simulation (see Table 7). While the networks are able to correctly differentiate between sentences drawn from the two grammars when ABA test patterns have a familiar shape (i.e., peaks), the networks cannot classify the test sentences when the ABA test sentences have a different form (i.e., valleys) than the one used during training (i.e., peaks) (see Table 7). The same behaviour was noticed in the newer Shultz & Bale's (2001) model (see Table 8). This discovery is important because it undermines Shultz & Bale's claims that their model recognizes "a syntactic pattern", and has the "ability to learn multiple syntactic forms simultaneously" (Shultz & Bale, 2001).

[25] The valleys are opposite to peaks: the numerical representations of the "B" syllables are lower than the numerical representations of the "A" syllables.

It is noteworthy that the number of test sentences used in these peaks vs. valleys experiments is the same as the number of test sentences employed by Shultz (1999), and Shultz & Bale (2001).

Table 7 Average network error when testing with "peaks" and "valleys" on Shultz' (1999) model

Experiment	Test patterns		Network error
Testing "peaks"	Grammatical	6 7 6	1. 21
		4 5 4	
	Ungrammatical	7 6 6	3.29
		5 4 4	
Testing "valleys"	Grammatical	7 6 7	4.55
		5 4 5	
	Ungrammatical	7 6 6	3.29
		5 4 4	

Table 8 Average network error when testing with "peaks" and "valleys" on Shultz & Bale's (2001) model

Experiment	Test patterns		Network error
Testing "peaks"	Grammatical	-5.0 5.0 -6.0 6.0 -5.0 5.0	13.36
		-4.0 4.0 -5.0 5.0 -4.0 4.0	
	Ungrammatical	-6.0 6.0 -5.0 5.0 -5.0 5.0	47.07
		-5.0 5.0 -4.0 4.0 -4.0 4.0	
Testing "valleys"	Grammatical	-6.0 6.0 -5.0 5.0 -6.0 6.0	55.64
		-5.0 5.0 -4.0 4.0 -5.0 5.0	
	Ungrammatical	-6.0 6.0 -5.0 5.0 -5.0 5.0	47.07
		-5.0 5.0 -4.0 4.0 -4.0 4.0	

4.5.3 Simulation of Gomez & Gerken's (1999) experiments

To further prove the conclusion that this model cannot robustly learn "syntactic patterns" (Shultz & Bale, 2001), I tested it with more complex grammars, like those used by Gomez & Gerken (1999) (see Figures 5 and 6).

As specified earlier, Gomez & Gerken (1999) performed four experiments on 1-year-old infants involving context-free grammars. They showed that infants acquired

specific information about the training grammar as demonstrated by their ability to discriminate new grammatical sentences from those with illegal endpoints (experiment 1), or with internal pair-wise violations (experiments 2 and 3), and, most importantly, the infants were able to discriminate between novel grammatical and ungrammatical sentences that were entirely generated with a new vocabulary (experiment 4). Only the fourth experiment required infants to generalize beyond the training vocabulary. I performed all these four experiments on Shultz & Bale's model, using the same finite-state grammars as Gomez & Gerken (1999). However, because of the particularity of Shultz & Bale's network structure, which only allows input sentences of the same length, I used a structure of 12 input and 12 output units, which is suitable for input sentences of 6 words (each word is a two-letter syllable). Since only a subset of Gomez & Gerken's input sentences have 6 words, I generated new 6-word sentences with the two grammars[26]. The model managed to replicate the infants' results on the first three experiments (where it was not required to generalize outside the set of the training patterns), but failed on experiment 4 (where generalization beyond the training set was necessary).

This demonstrates once again that Shultz & Bale's model is not able to respond correctly to more general or complex stimuli. Granted, the results reported by Shultz (1999), and Shultz & Bale (2001) do replicate the infants' results from Marcus *et al.*'s study (1999), and both models represent genuine connectionist implementations of the

[26] In the original Gomez & Gerken experiments, only 4 of the 10 input sentences generated with the first grammar (Figure 5), and 5 of the 10 input sentences generated with the second grammar (Figure 6) contain 6 words (the other sentences contain 3, 4, and 5 words). In the simulation of Gomez & Gerken experiments with Shultz & Bale's model, I added 6 new 6-word sentences formed with grammar #1, and 5 new 6-word sentences formed with grammar #2. My simulations thus use 10 6-word training sentences, 10 novel 6-word test sentences formed with the familiar grammar, and 10 6-word test sentences formed with the unfamiliar grammar.

Marcus *et al.*'s work. I believe, however, that these models mostly (if not entirely) rely on non-essential characteristics of the input patterns (e.g., numerical shape), rather than their grammatical structure. The training algorithm is not strong enough and does not go deep enough in order to induce the learning of abstract categories that ensures a robust discrimination of the input patterns and also, very importantly, generalization to novel items. This conclusion has been proved during my knowledge representation analysis on this model.

4.6 Knowledge Representation Analysis

In order to explain the behaviour of their model, Shultz & Bale (2001) employ three different analysing techniques: connection weight analysis, principal component analysis (PCA) of contributions, and hidden unit activation analysis.

4.6.1 Connection weight analysis

Within this analysis, Shultz & Bale (2001) display the connection weights of one network following ABA training in experiment 1. In this training condition, the network generates two hidden units. Shultz & Bale found that the connection weights between the first hidden unit and those output units that correspond to the first and third words (the "A" words) of a sentence are very similar to each other, and, at the same time, are significantly higher than the connection weights between the same first hidden unit and those output units that correspond to the middle word (the "B" word). Also, Shultz & Bale found that the connection weights between the second hidden unit and those output units that correspond to the "B" words are significantly higher than the weights between the same second hidden unit and those output units that correspond to the "A" words.

Based on these findings, Shultz & Bale claim that their model learns the duplicate-word category (i.e., the "A" word for ABA training) in the first hidden unit, and the single-word category (the "B" word) in the second hidden unit.

Although the connection weights of the network shown by Shultz & Bale (2001) seem to indicate that the network may learn the category of the two types of words, I found that this is frequently not the case. In my peaks vs. valleys experiment, in more than 25% of the networks that I trained (5 out of 16 networks that were trained in the peaks vs. valleys experiment), the hidden nodes are not reliable category recognizers. Those networks develop connection weights that do not conform to the Shultz & Bale's account (see Table 9): there are no significant differences among connection weights for both the first and the second output units to indicate that the network learns the category of the two types of words. The connection weights shown in Table 9 pertain to one such network but are very similar among all five networks that exhibit this behaviour.

Table 9 The connection weights of a Shultz & Bale network trained in the peaks vs. valleys experiment

	Output Unit #1	Output Unit #2	Output Unit #3	Output Unit #4	Output Unit #5	Output Unit #6
Hidden Unit #1	-4.56	4.56	-8.42	8.42	-4.56	4.56
Hidden Unit #2	-2.04	2.04	2.58	-2.58	-2.04	2.04

4.6.2 PCA of contributions

For each input pattern, Shultz & Bale record the activations of both hidden nodes, and the connection weights between the hidden and output units (i.e., the output weights). Then, they compute the network contributions for each input pattern, which are the

products between hidden activations and output weights, and generate a so-called contribution matrix, where each row in the matrix represents the network contributions of one input vector. The contribution matrix is transformed to covariance form[27] and principal component analysis (PCA) is performed on the covariance matrix. For one ABA-trained network in experiment 1, Shultz & Bale argue that PCA reveals two principal components: the first principal component represents the variation of contributions caused by the sonority sums (the arithmetic sum between the sonority values of consonants and vowels) of the single word-category (i.e., the "B" word), whereas the second component represents the variation of contributions caused by the sonority sums of the duplicate word-category (i.e., the "A" word). Based on these findings, Shultz & Bale claim that their model learns to encode the input stimuli as whole words (consonant-vowel combinations), rather than as separate, individual letters. Figure 24 displays the contributions of a network trained in experiment 1, along with the sonority sums of both the "A" and "B" words that form the input sentences.

[27] The covariance matrix consists of the variances of contributions corresponding to each output unit, along the main diagonal, and the covariances between contributions corresponding to each pair of output units, in the other matrix positions. The variance of a set of values is the arithmetic average of the squared distance from the mean of those values, whereas covariance provides a measure of how strongly correlated two sets of values are.

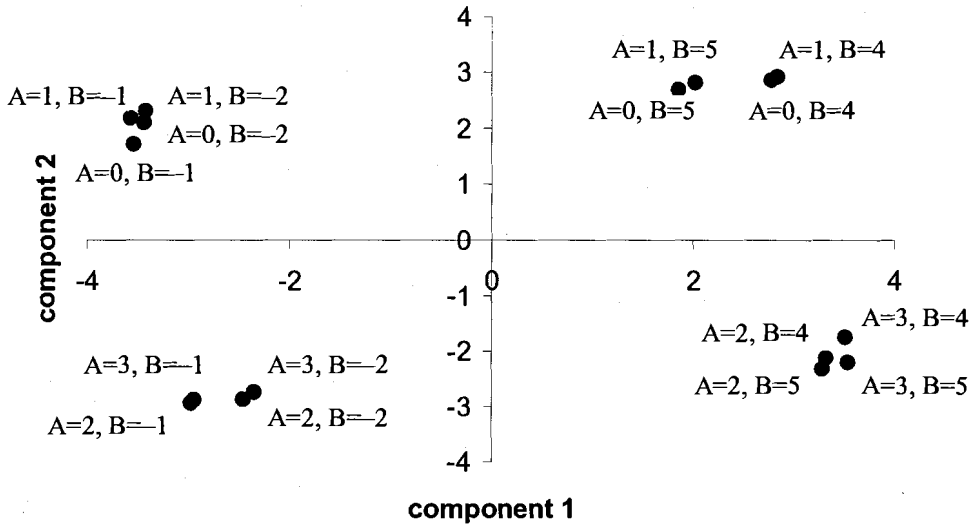

Figure 24 Projections of network contributions onto the first two principal components of an ABA-trained network in experiment 1. The labels show the sonority sums of the A and B words which form the input patterns that generate these contributions. The figure indicates possible dependencies between network contributions and sonority sums of B words (along the first principal component), and between network contributions and sonority sums of A words (along the second principal component).

According to Figure 24, as Shultz & Bale claim, it is possible to argue that the variation in network contributions is caused by the *sonority sums* of the input syllables. However, as shown in Figure 25 (which displays *exactly* the same network contributions, but shows just the sonority values of *consonants* contained in the "A" and "B" words that form the input sentences), it can be argued that the variations in network contributions are rather caused by the individual sonority values of the consonants that appear in the "A" and "B" words: the first principal component reflects the variation in contributions caused by the sonority values of consonants in the "B" words, whereas the second component reflects the variation in contributions caused by the sonority values of consonants in the "A" words. Apparently, because there is so little variation in the vowel

representations (all vowels in the input set are represented by a sonority value of either 4.0 or 6.0), they do not have a significant impact on the network contributions.

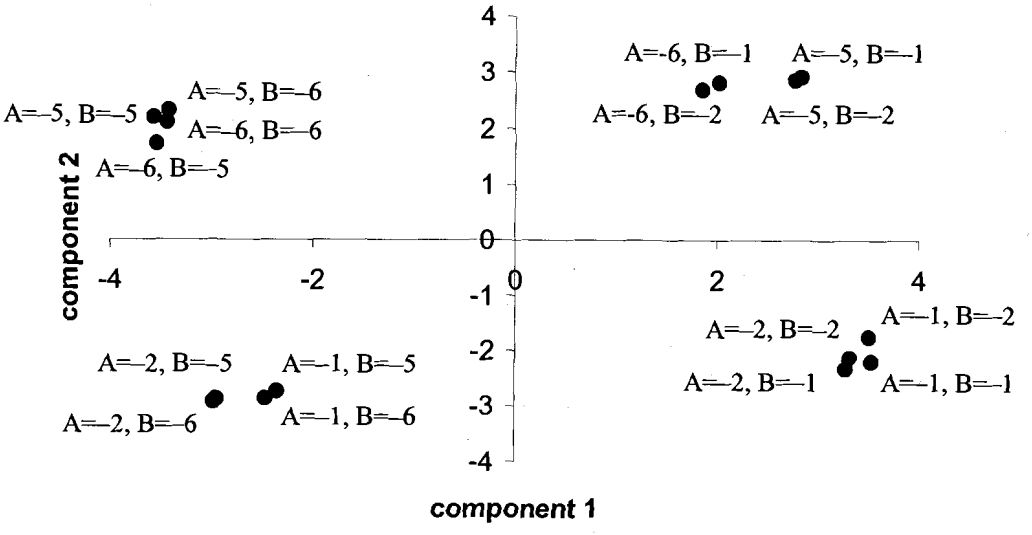

Figure 25 Projections of network contributions onto the first two principal components of an ABA-trained network in experiment 1. The labels show the sonority values of consonants in the A and B words that form the input patterns that generate these contributions. The figure indicates dependencies between network contributions and consonant values within B words (along the first principal component), and between network contributions and consonant values within A words (along the second principal component).

Similarly, in experiments 2 and 3 (where the input stimuli have changed to eliminate the differences between phonetic features that occur in the first experiment), the variation in network contributions is *again* dictated by *individual sonority values of consonants*. Figures 26 and 27 display the same projections of contributions onto the first and second principal components of a network trained with ABA patterns in experiment 2. The labels in Figure 26 show the *sonority sums* of the "A" and "B" words, whereas the labels in Figure 27 show the sonority values of the *consonants* contained in the "A" and "B" words.

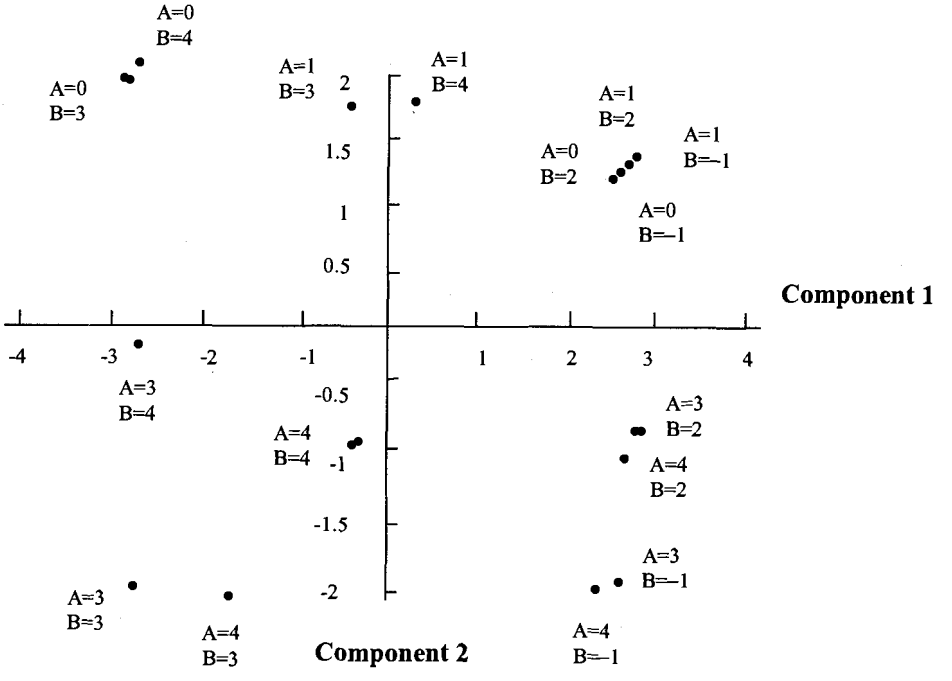

Figure 26 **Projections of network contributions onto the first two principal components of an ABA-trained network in experiment 2. The labels show the sonority sums of the A and B words which form the input patterns that generate these contributions. The figure indicates possible dependencies between network contributions and sonority sums of B words (along the first principal component), and between network contributions and sonority sums of A words (along the second principal component).**

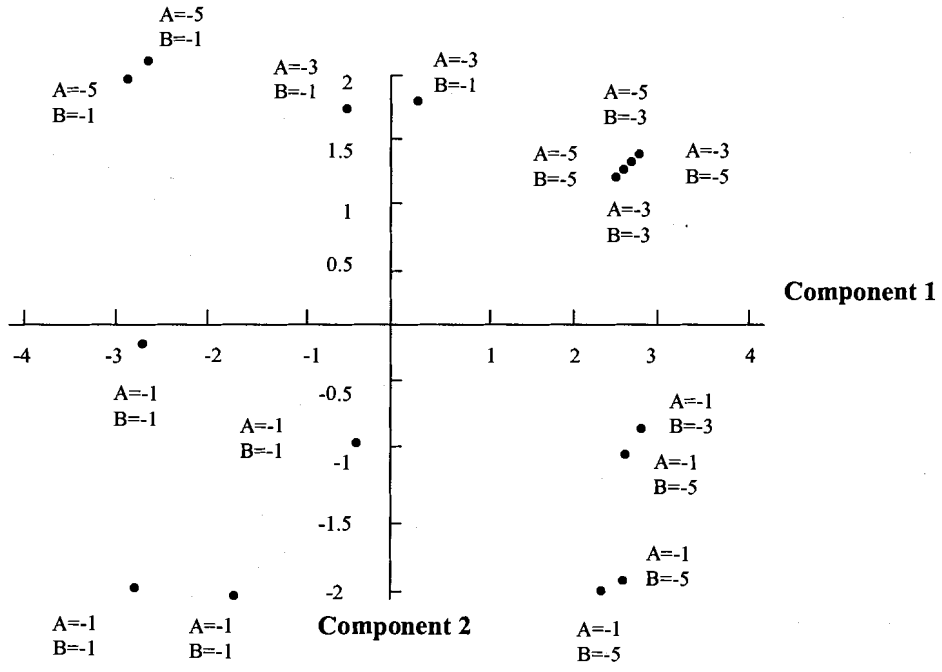

Figure 27 Projections of network contributions to the first two principal components of an ABA-trained network in experiment 2. The labels show the sonority values of consonants in the A and B words that form the input patterns that generate these contributions. The figure indicates dependencies between network contributions and consonant values within B words (along the first principal component), and between network contributions and consonant values within A words (along the second principal component).

According to Figure 26, there seems to be a tendency of contributions to form several clusters with regard to the sonority sums of "A" and "B" words: the first principal component reflects the variation caused by the sonority sums of "B" words, whereas the second principal component reflects the variation caused by the sonority sums of "A" words. Therefore, as Shultz & Bale argue, it is possible to infer that the variation in contributions is caused by sonority sums. However, based on Figure 27, in reality, the variation in contributions is dictated by the *individual sonority values of consonants* that occur in the "A" and "B" words: the first principal component reflects the variation caused by the sonority values of consonants in "B" words, whereas the second

component reflects the variation caused by the sonority values of consonants in "A" words. The sonority values of vowels do not have a significant impact on the network contributions. It is the consonants that have the greatest influence on network contributions. This suggests that, contrary to Shultz & Bale's claims, the network does not learn to encode the input stimuli as whole syllables, but as separate, individual sonority values.

4.6.3 Hidden unit activations

The final analysis performed by Shultz & Bale (2001) involved the values of hidden activations. For a network trained with ABA patterns in experiment 1, and another network trained with ABB patterns in experiment 3, Shultz & Bale discovered a negative correlation between the activation of the first hidden unit and the sonority sums of the duplicate word ("A" words in ABA training, and "B" words in ABB training). At the same time, there is a positive correlation between the activation of the second hidden unit and the sonority sums of the single word ("B" words in ABA training, and "A" words in ABB training). Shultz & Bale claim that this is yet another proof that the first hidden unit learns to encode the *sonority sum* of the duplicate word, whereas the second hidden unit learns to encode the *sonority sum* of the single word.

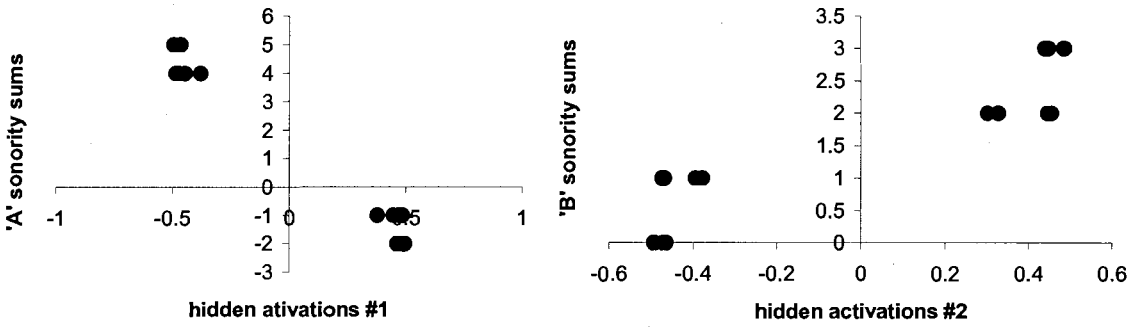

Figure 28 **The correlation between the sonority sums of A and B words and the hidden activations of an ABA-trained network in experiment 1. There seems to exist a negative correlation between the activations of the first hidden unit and the sonority sums of the A words, and a positive correlation between the activations of the second hidden unit and the sonority sums of the B words.**

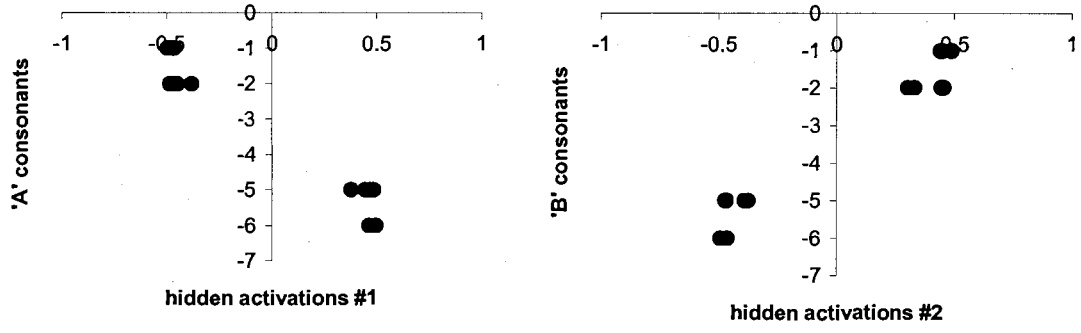

Figure 29 **The correlation between the sonority values of the consonants in A and B words and the hidden activations of an ABA-trained network in experiment 1. There exists a negative correlation between the activations of the first hidden unit and the sonority values of the consonants in A words, and a positive correlation between the activations of the second hidden unit and the sonority values of the consonants in B words.**

For a network trained with ABA patterns in experiment 1, Figure 28 displays the relation between the hidden activations and the sonority sums of the input words, and

Figure 29 displays the relation between the same hidden activations and the sonority values of *consonants* contained in the input words. According to Figure 28, it can be argued that there is a positive/negative correlation between hidden activations and the sonority sums of the input words. However, Figure 29 shows that these correlations are actually caused by the sonority values of the *consonants* that are contained in the input words. The sonority values of the vowels contained in the input words barely affect the correlation between the hidden activations and the input syllables (see Figure 30). It is the consonants that have the greatest influence on how the network's internal representations are generated. This suggests that, contrary to Shultz & Bale's claims, the network does not learn to encode the input stimuli as whole syllables (consonant-vowel pairs). Instead, the network encodes the separate, individual sonority values that appear in the input patterns.

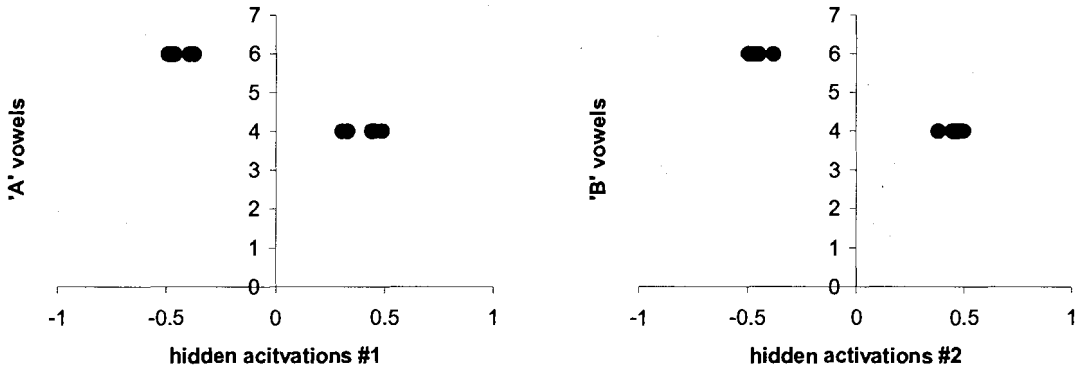

Figure 30 The correlation between the sonority values of the vowels in A and B words and the hidden unit activations of an ABA-trained network in experiment 1. There is no difference in the way the sonority values of the A and B *vowels* correlate with the hidden activations.

5 ALTMANN & DIENES', AND ALTMANN'S MODELS

Altmann & Dienes' (1999) work is based on a previous model of their own: Dienes, Altmann, & Gao (1999). That earlier model incorporates a modified simple recurrent network that can "transfer its knowledge of artificial grammars across domains" (Dienes, Altmann, & Gao, 1999). Later, Altmann & Dienes (1999) adapt that model to simulate Marcus *et al.*'s (1999) experiments on infants. More recently, Altmann (2002) further refines Altmann & Dienes' (1999) model by adding a preliminary training phase, and modifying the test procedure.

5.1 Network Architecture and Preliminaries

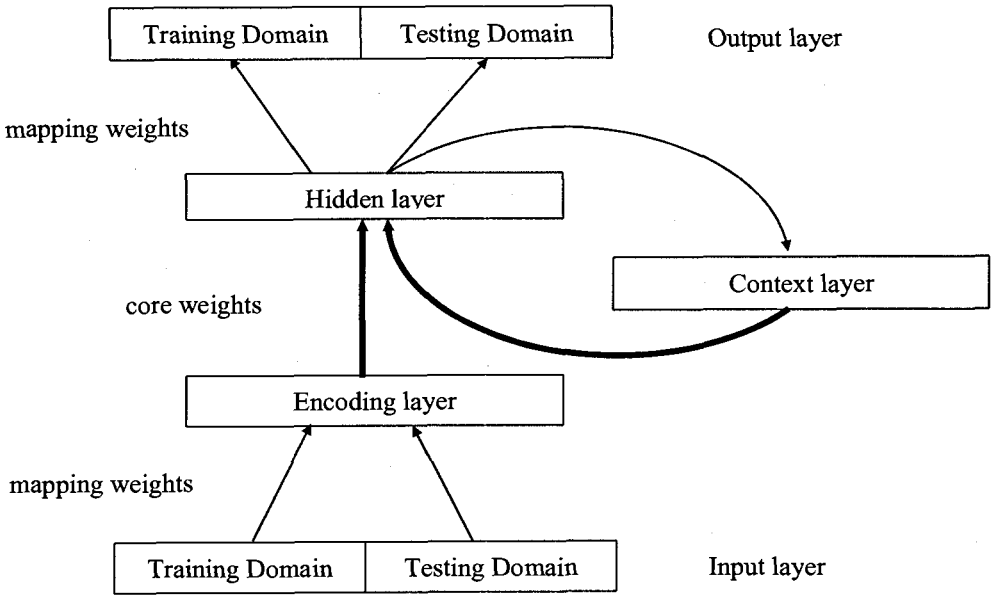

Figure 31 Altmann & Dienes' (1999) model: a SRN with an extra encoding layer

Altmann & Dienes' model employs a simple recurrent network with an additional layer of units between the input and hidden layers of the SRN. This additional layer is used to re-encode the input representations of two domains (the training and test domains). The function of this extra layer is to provide an abstract, common encoding of two input sets (see Figure 31).

Connection weights between the encoding and hidden layers, as well as between the context and hidden units, are called "core weights"; they are frozen after the initial training. All other connection weights are called "mapping weights", and are allowed to change even during testing, while the second input set (the test domain) is learned.

5.2 Training Procedure and Input Representation

Training is performed using the back-propagation algorithm, similar to other simple recurrent networks. However, unlike other simple recurrent networks, this model contains one extra layer between the input and the hidden layers (the encoding layer). This new layer introduces another set of training weights (between the input and the encoding layers), which are also adjusted with the back-propagation algorithm. The initial value of all connection weights is chosen randomly from a uniform distribution between –0.5 and +0.5. All input vectors are completely orthogonal: just one input unit is active at any time, corresponding to a given syllable (localist representation). Each sentence consists of three syllables, created in conformity to ABA and ABB grammars. Sentences are presented to the network one syllable at a time, starting with the activation of a special "start" unit and concluding with the activation of the "end" unit. Therefore, if there are n syllables, the input layer will have $n + 2$ units. The input layer consists of two parts, corresponding to the two domains: the units that are part of the first domain are

only activated during training (the domain delimited by the training syllables), while the input units that are part of the second domain are only activated during testing (this domain is delimited by the test syllables) (see Figure 31). The same pattern of units exists for the output layer as well.

During training, the units in the first domain are activated (activation is set to 0.9 for the unit that corresponds to the current syllable, all other units having an activation of 0.1). The network is trained to predict the next syllable in the sentence. Therefore, the target activation on the output layer is 0.9 for the unit associated with the next syllable in the current sentence, and 0.1 for all other syllables.

Altmann & Dienes (1999) use this model to simulate Marcus *et al.*'s (1999) study on infants. They train eight instances of this network on 16 sentences formed with the ABA grammar and eight other networks on 16 sentences formed with the ABB grammar.

5.3 Testing and Results

During testing, for each test stimulus, the "core weights" are frozen, and only the "mapping weights" are changed for a number of iterations, until the network has learned the encoding of the test item. The mapping weights start at arbitrary random values. After this additional learning process, all connection weights are finally frozen, and the network is tested on the current test pattern. The procedure is repeated for all patterns in the test set. Although this training/testing procedure may seem biologically implausible, Almann & Dienes argue it mimics an adaptive learning mechanism, where the learning rate gradually decreases while the learning progresses.

In order to support their assumption, Altmann & Dienes argue that it is generally believed that humans' ability to learn artificial grammars steadily improves over hours of exposure (Mathews *et al.*, 1989), and therefore "it seems unlikely [humans] do spontaneously freeze their weights just due to the passage of a few minutes learning" (Dienes *et al.*, 1999). Furthermore, Altmann & Dienes argue that the network is not able to generalize to novel items if the connection weights are not frozen. According to Altmann & Dienes, this inability is caused by the so-called catastrophic interference (McCloskey & Cohen, 1989), which, in essence, means that novel input causes the unlearning of previously learned items. They claim that catastrophic interference can be reduced by interleaving the learning of different items (Hetherington & Seidenberg, 1989), and that the human brain may solve this problem by using two complementary systems (McClelland, McNaughton, & O'Reilly, 1995): neocortex and hippocampus. According to McClelland *et al.* (1995), the hippocampus "continually reinstates new and old memories so as to integrate them into the structured neocortical memory system" (Dienes *et al.*, 1999). McClelland *et al* (1995) argue that the neocortical system is responsible for abstracting the common structure of a set of items, and, therefore, needs to be protected against catastrophic interference by interleaving the input items so that novel input does not cause the forgetting of previously learned input. Altmann & Dienes claim that they simulate the two systems of the brain within their simple recurrent network. Catastrophic interference is avoided by freezing the core weights (which resemble the neocortex) when the network is presented with novel items (the mapping weights resemble the hippocampus). This way, Altmann & Dienes argue that the previously learned items are not forgotten by the network during testing.

In order to simulate Marcus *et al.*'s (1999) study, during testing, Altmann & Dienes present 2 ABA and 2 ABB test sentences to the network, in random order, using input nodes that were not used during the first phase of training. Altmann & Dienes measure their results by computing the cosine of the angle between the vector of the actual syllables in each position of the sentence, and the vector of predicted syllables in each position. Altmann & Dienes report that, for a learning rate of 0.5 and momentum of 0.01, and 10 iterations around each test syllable, that angle is smaller for familiar sentences (test sentences that have the same form as the training patterns) than for unfamiliar ones, and the Euclidian distance between prediction and target is smaller for familiar sentences than for unfamiliar ones. Therefore, they claim that "like the infants studied by Marcus *et al.*, our networks successfully discriminated between the test stimuli", and "the conclusions by Marcus *et al.* stated in the report are premature" (Altmann & Dienes, 1999).

5.4 Marcus' Evaluation

Despite this report, Marcus is not convinced that Altmann & Dienes' model is a counter-example to his claims. He sees three reasons why this model does not provide a correct account of his results:

- The interpretation of network outputs is questionable. Marcus believes that the network outcome is highly dependent on the way the interpretation is performed. According to Marcus, the most common way to interpret a connectionist model is in terms of "the most active output unit at any time" (Marcus, 1999), and he argues that if Altmann & Dienes' model is interpreted in this way, "one finds that the model does not

learn the training grammar, but instead oscillates between (say) the ABA and ABB grammars" (Marcus, 1999).

I agree that the way one interprets a connectionist model is subjective. However, I think that Marcus' concern is exaggerated. Although interpreting a connectionist model in terms of the most active output unit at any given moment is indeed very common, there are other ways to measure network's performance depending on the task being performed and on the output representation. Although computing the cosine of the angle between the actual and target vectors is less common, it has been used in other connectionist studies, and it does make sense in this classification task. The fact that Altmann & Dienes' evaluation method is not typical should not be an issue of concern.

- There exists an external mechanism that can freeze a subset of connection weights, while iterating through test items. Marcus argues that this technique is not generally used in connectionist simulations, limiting this model's generality and plausibility. He states "it is unclear what sort of neural system could implement this in the brief period of time which the infants have in our experiments" (Marcus, 1999).

I agree that this issue is problematic. Altmann & Dienes want to simulate an adaptive learning technique, in which the learning rate decreases while training progresses. The adaptive technique is based on the assumption that human learning is a continuous and lengthy process, without any sudden freeze of neural connections. Nevertheless, the way Altmann & Dienes carry out this mechanism is questionable. I agree with Marcus that freezing only a subset of the connection weights, while others remain trainable, is arbitrary and not biologically plausible. Later, Altmann (2002) proposed a slightly changed version of the same model that eliminates the need for the

partial freezing of the connection weights. However, as I argue below, this new model has its own problems.

- The model just maps the encoding of one set of words onto the encoding of another set of words. Marcus performed a simple experiment on Altmann & Dienes' model: he habituated the model on 16 ABA sentences, and then tested it on the novel sentence *wo fe wo*. Marcus noticed that after a few iterations, the model mapped *wo* and *fe* into two training syllables (*ga* and *ti*). Subsequently, he tested the same network on two other sentences: *fe wo wo* and *fe wo fe*. Marcus reports that, according to the technique used by Altmann & Dienes to measure the network's outputs, the angle between the target and prediction was smaller for *fe wo wo* than for *fe wo fe*, because the model was already familiarized with *wo* appearing as the third word in the sentence. In other words, the model incorrectly favoured a sentence generated with the unfamiliar grammar. In a subsequent experiment on infants (Marcus & Bandi Rao, 1999), Marcus showed that the infants actually looked longer at *fe wo fe* than *fe wo wo*, i.e., they correctly favoured the sentence generated with the familiar grammar.

The point raised here by Marcus is very important, because it suggests that Altmann & Dienes' model is fundamentally flawed, and behaves differently than how the authors claim. My own investigation of this model proved that.

5.5 Personal Investigation

In recent work (Vilcu & Hadley, 2003), we performed the same kind of experiments as Altmann & Dienes (1999), using the same network architecture, training algorithm, and input data, on double the number of networks. We trained 16 networks

with ABA sentences, and another 16 with ABB patterns. We used both our own simulator (VNNS), and the PDP++ neural network simulator. There are no significant differences between the results reported by VNNS and those reported by PDP++.

We discovered that the model behaves inconsistently, and that the Euclidian distance between target and actual outputs is always smaller for ABA test sentences, regardless of the training grammar. I find this incompatible with Altmann & Dienes' claims that their network "can model aspects of Marcus *et al.*'s data" (Altmann & Dienes, 1999). I also tried various other learning parameters (learning rate, momentum, number of iterations, initial weights), and even more complex grammars (I repeated all four experiments of Gomez & Gerken using the finite-state grammars showed in Figures 5 and 6). In each case my results show that Altmann & Dienes' results are not at all robust. None of the sixteen separate networks that I trained in these conditions generated results compatible with Altmann & Dienes' statements; in each case, the networks were unable to differentiate between familiar and unfamiliar test sentences. Therefore, I believe that Altmann & Dienes' claim that "like the infants (...), our networks successfully discriminated between the test stimuli" (Altmann & Dienes, 1999) is not sufficiently demonstrated. As discussed later in this chapter, my knowledge representation analysis performed on Altman & Dienes' model shows that this network does not thoroughly learn the sequential structure of the input sentences. Instead, it extracts non-useful information, such as the difference among the middle words of each sentence, which does not help the network differentiate between the two grammatical structures.

5.6 Altmann's Model

More recently, Altmann (2002) employed a variation on the Altmann & Dienes' (1999) experimental design. He made the following two changes to the original model:

- Eliminated the partial freezing of the connection weights during testing.
- Added a preliminary training phase.

Altmann argues that the freezing of the connection weights that exists in Altmann & Dienes (1999) may not be required if it is replaced with an equivalent mechanism that partitions the internal representational space of the network in such a way that makes the "unlearning" of previously trained items very difficult. Altmann claims that this can be achieved by densely populating the internal representational space using a pre-training phase that employs an "*arbitrary* grammar and associated vocabulary" (Altmann, 2002) (my emphasis). According to Altmann, the expected effect of this pre-training phase is that whenever the network needs to form a new internal representation as a result of a novel input item, this new representation will not replace any previously learned representations. Instead, it would "integrate with existing, previously entrenched, representations" (Altmann, 2002).

5.6.1 Training and test procedures

During the preliminary training phase, Altmann chose to use the same training procedure as in an earlier work by Elman (1990). According to this procedure, 252 different sentences are generated using two very simple grammars such as *Noun-Verb*, and *Noun-Verb-Noun*. For example: *man see book*, *woman eat sandwich*, *rock break*, etc.

Twenty-nine words are used in pre-training, each word being represented by one unit in the input layer (localist representation), and a ten thousand sentence pre-training corpus (approximately 40 repetitions of each individual sentence) is generated. This corpus is presented six times to 16 different networks. Following pre-training, the networks are trained to differentiate between two simple grammars (similar to the task performed by the infants in the Marcus *et al.* study, 1999): 8 networks are trained on 16 ABA sentences and the other 8 are trained on 16 ABB sentences. The training sentences require 8 new words that are encoded on 8 new input units (that were not used in pre-training). The testing is performed with 4 novel sentences (2 ABA and 2 ABB sentences) that used 4 new words encoded on 4 other input units. The model is trained to output the next word (syllable) in the current sentence.

5.6.2 Results

To measure the networks' performance, Altmann calculated the product moment correlation between the target output vectors and the networks' actual predictions. According to these calculations, Altmann stated that the networks were better at predicting sentences that were generated with the familiar (training) grammar. Thus, his model would count as an example of a connectionist model that successfully reproduces the results reported by Marcus *et al.* (1999).

However, as Altmann acknowledges, one possible reason for this result could be the fact that the pre-training corpus contained sentences that had an underlining ABA structure (for example, *cat chase cat*), which could facilitate the learning of that grammar. Altmann is aware of this possible problem and repeats his experiment by eliminating all those sentences from the pre-training corpus. He claims that the new

results prove his initial assertion that the model is indeed able to differentiate between the two syntactic structures.

5.6.3 Personal investigation

Altmann states that the grammar used in the pre-training phase is completely arbitrary, and that the only reason for doing this initial phase of training is to populate the internal representational space of the networks densely enough so that any new representation is mapped into an existing one. According to this theory, the choice of a particular pre-training grammar is not relevant. However, by doing my own experiments on this model, I found that this is not the case. When I replaced the pre-training grammar with a more complex one (like the finite-state grammar shown in Figure 5), I discovered that the networks *fail to discriminate between the two syntactic structures*. This proves that, contrary to Altmann's claims, the structure of the pre-training patterns is very important for the success of the networks.

Granted, the model does mirror the results reported by Marcus *et al.* (1999), and I was able to reproduce Altmann's results in my own experiments. However, in doing so, I found that the model seems to rely on a very particular structure of the pre-training patterns, which facilitates the learning of these two simple grammars. As mentioned above, during the preliminary training phase, Altmann uses preliminary training sentences having the form *Noun-Verb*, and *Noun-Verb-Noun*. When presenting these patterns to the network, they can form underlying ABA or ABB structures that can help the network learning those simple grammars. For example, two consecutive sentences *man see car* and *car break* form an underlying ABB pattern (*see car car*). By analysing the entire pre-training corpus, I discovered that roughly 10% of the corpus form

underlying ABA and ABB patterns. When I eliminated those patterns from the pre-training corpus, and repeated the experiments, the model failed to differentiate between the ABA and ABB structures. Alternatively, if the pre-training is performed with sentences generated by a different grammar (e.g., the context-free grammar shown in Figure 5), the model is still unable to learn the two ABA and ABB structures.

With the original pre-training corpus (and grammar) in place, I discovered that the networks could not learn more complex grammars. I repeated all four experiments of Gomez & Gerken (1999), using the finite-state grammars shown in Figures 5 and 6, and found that none of the 16 networks that I trained was able to successfully replicate the results reported by Gomez & Gerken (1999). As my knowledge representation analysis shows, this model is unable to discover the sequential structure of the input sentences, not even when those sentences are formed with the simpler grammars. It focuses instead on non-essential and very particular characteristics of the input patterns (such as discovering the various consonants and vowels in the training stimuli), which do not help the network differentiate between two grammatical structures.

5.7 Knowledge Representation Analysis

5.7.1 Altmann & Dienes' model

The analysis of Altmann & Dienes' model contains both principal component analysis (PCA), and hierarchical cluster analysis. Unless specified otherwise, all hidden activation vectors used in these analyses were collected at the end of each input sentence, and the networks were trained with ABA patterns. Since the input representation is localist, 4 of the input units act as "A" words (denoted "A" word #1, "A" word #2, "A"

word #3, and "A" word #4), and other 4 units represent "B" words (denoted "B" word #1, "B" word #2, "B" word #3, and "B" word #4).

5.7.1.1 Principal component analysis

PCA of hidden activations for a network trained with ABA patterns reveals many principal components, reflecting a lot of variation in the internal representations (the first 6 components account for about 85% of the variation). The projection of hidden activations onto the first 4 principal components (which account for about 75% of the variation) reveals a fairly complex representation of those activations. According to the first two principal components (see Figure 32), the network's internal representations at the end of each sentence show a dependency on the "B" words (i.e., the middle words in ABA training)[28] of each sentence. Based on Figure 32, there is a certain tendency of the internal representations to form several clusters with regard to the "B" words that appear in the input sentences, but these clusters are not very well formed and separated. Although the internal representations formed by sentences that contain three middle words ("B" word #1, #2, and #4) are quite distinctive, the internal representations formed by sentences that contain "B" word #3 crosscut other representations.

[28] With regard to the words at the end of each sentence, the "B" words are the previous words (considering ABA training).

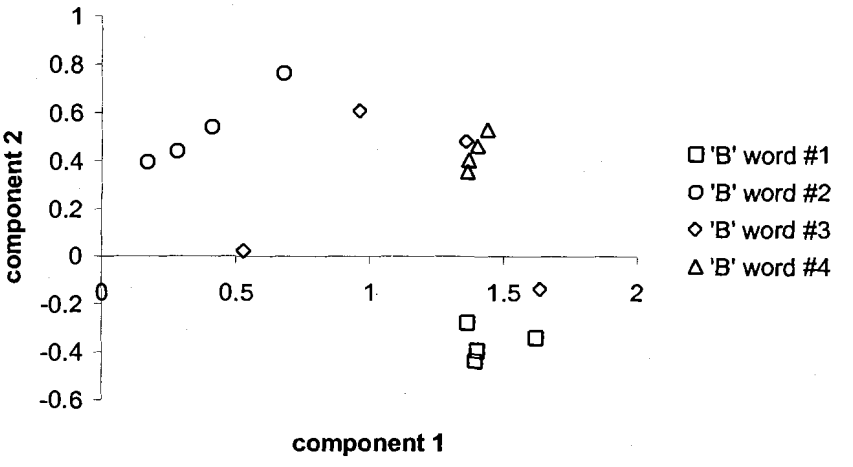

Figure 32 Projections of hidden activations at the end of each training sentence onto the first two principal components (with regard to the middle words in each sentence). There is a certain tendency of internal representations to group based on the "B" words in the middle of each training sentence.

Figure 33 displays the projections of hidden activations at the end of each sentence onto the third and fourth principal components. The projections are shown with regard to the "A" words that appear at the end of each sentence. According to Figure 33, there is a weak tendency for some of these projections to form several groups, but there is a high degree of overlap, especially between activations formed by the "A" word #2, and those formed by the "A" word #3.

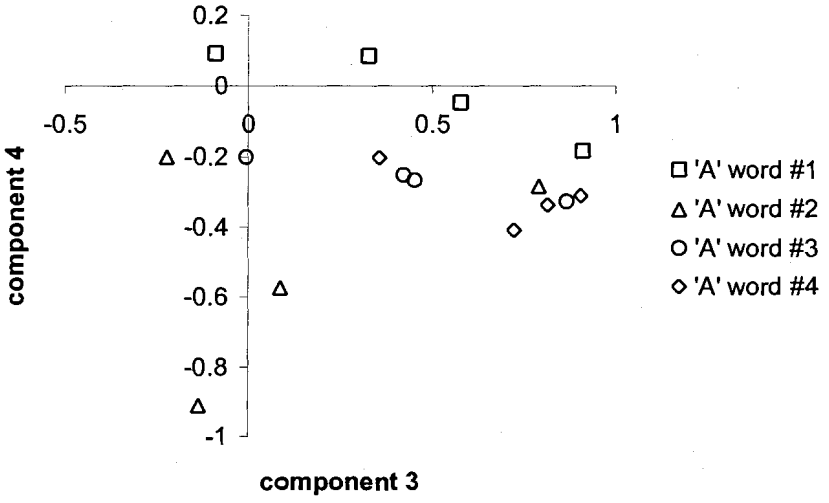

Figure 33 Projections of hidden activations at the end of each training sentence onto the third and fourth principal components (with regard to the "A" word in each sentence). There are some weak tendencies of internal representations to group based on the "A" words at the end of each sentence.

Although Figures 32 and 33 reveal that there are some (weak) tendencies within networks to group the internal representations, these groupings are not very useful with regard to assisting the network differentiate between ABA and ABB patterns. For example, according to Figure 32, 75% of the network's internal resources are used to separate the input sentences based on the middle words that appear in the training sentences. Even if the model were completely successful in performing this separation, it would not help it in differentiating between those ABA and ABB test sentences that have the same middle word (e.g., *ga ko ga* vs. *ga ko ko*).

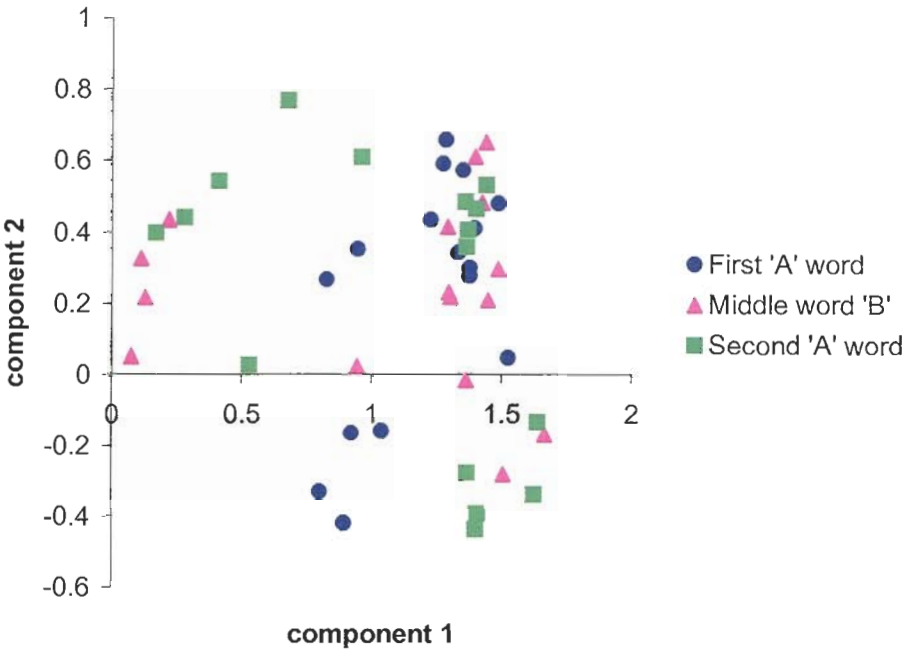

Figure 34 **Projections of hidden activations onto the first two principal components, formed by each input word in the training sentences (the first "A" word, the middle "B" word, and the last "A" word in each sentence are shown). There is a high degree of overlap among the internal representations formed by the first and second "A" words, and also among those formed by the middle "B" words.**

In order for this model to robustly distinguish between the ABA and ABB grammars, the network needs, I believe, to abstract the syntactic structure of input sentences. When trained with ABA patterns, the network should be able to detect the fact that the first and third words are the same, whereas the middle word is different. Figure 34 displays the projections of hidden activations onto the first two principal components for *each input word* of the training sentences. According to Figure 34, there is a high degree of overlap among words that appear in each of the three positions within

sentences. Apparently, the network is not able to extract the syntactical structure of the input patterns, and this may explain why it fails to generalize to novel input.

5.7.1.2 Hierarchical cluster analysis

In order to see how the model groups the test stimuli, a cluster analysis on hidden activations at the end of each *test* pattern is performed. Because, originally, there are only 2 grammatical and 2 ungrammatical test sentences, in order to increase the statistical significance of this analysis, 6 new test sentences were generated in each condition[29]. Figures 35-42 display the results of the cluster analysis performed on 16 hidden activations (8 grammatical and 8 ungrammatical) for 8 different ABA-trained networks. Since I am interested in seeing whether the networks are able to separate the hidden activations into two clusters, Figures 35-42 also display, in two different colours, how the networks form the two clusters, and how they associate the hidden activations to the two groups. According to Figures 35-42, none of the 8 networks is able to correctly separate all 16 test patterns into the grammatical vs. ungrammatical groups. There is an important level of overlap between the hidden activations formed by the ABA and ABB patterns (see Figures 35-42 for details), suggesting that the model cannot robustly separate the grammatical test sentences from the ungrammatical ones.

The following 8 graphs (Figures 35 to 42) display the results of cluster analysis performed on 8 different networks (denoted network #1 to network #8).

---

[29] The new test sentences are generated by permuting the words used in the original test sentences. For example, 4 test syllables are used in experiment 1: *wo*, *de*, *fe*, *ko*, and 4 initial test sentences: *wo fe wo*, *de ko de*, *wo fe fe*, and *de ko ko*. The 6 new ABA test sentences are: *wo ko wo*, *de fe de*, *fe wo fe*, *ko de ko*, *fe ko fe*, and *ko fe ko*. The 6 new ABB test sentences are: *wo ko ko*, *de fe fe*, *fe wo wo*, *ko de de*, *fe ko ko*, and *ko fe fe*.

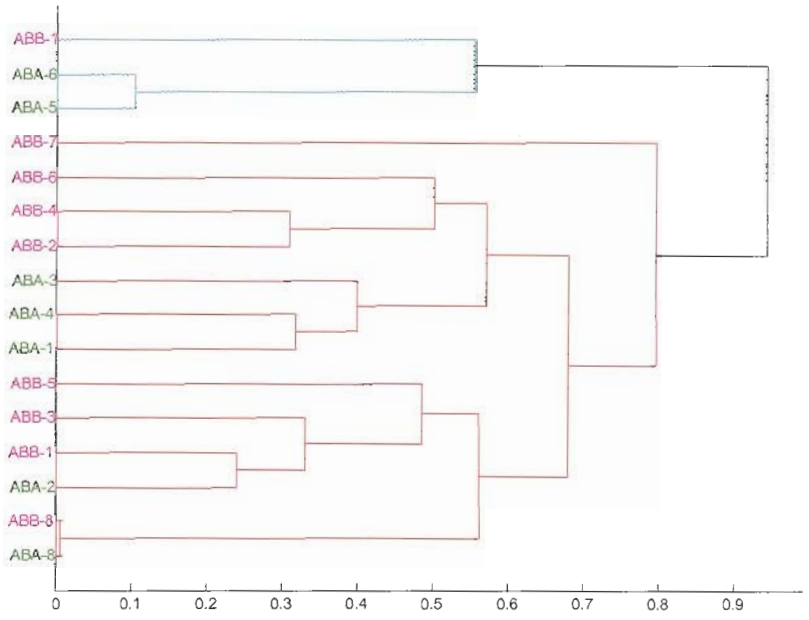

Figure 35 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #1 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 2 ABA and 1 ABB sentence (67% ABA), whereas cluster 2 (red) contains 6 ABA and 7 ABB test sentences (46% ABA).**

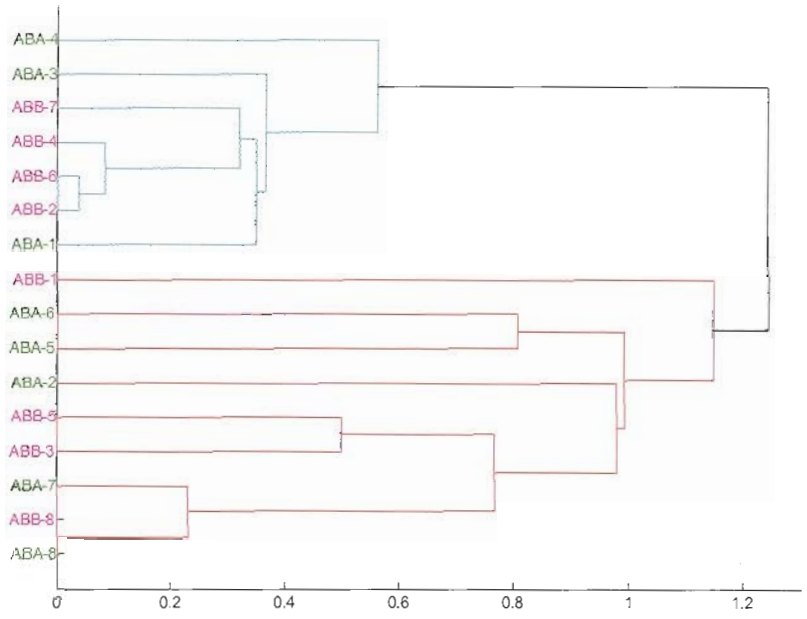

Figure 36 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #2 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 3 ABA and 4 ABB sentences (43% ABA), whereas cluster 2 (red) contains 5 ABA and 4 ABB test sentences (56% ABA).**

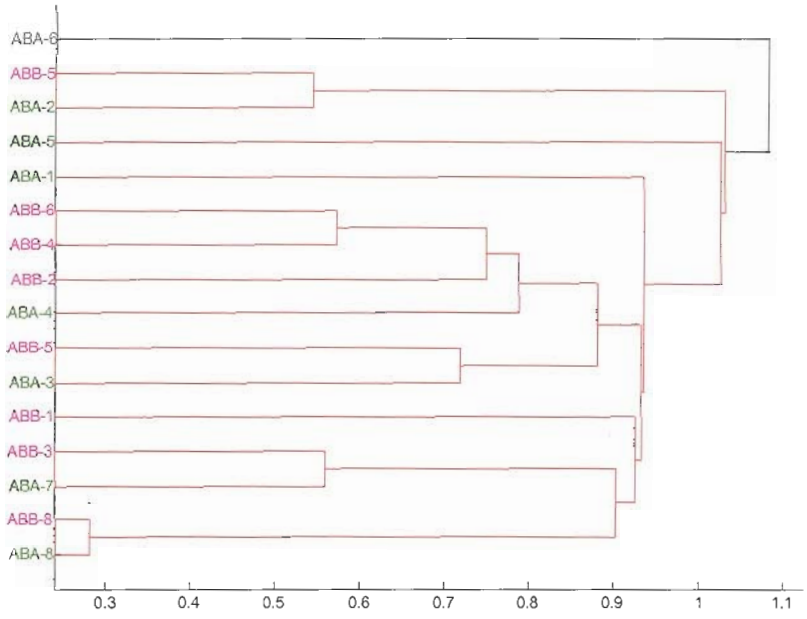

Figure 37 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #3 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 contains only 1 ABA sentence (100% ABA), whereas cluster 2 (red) contains 7 ABA and 8 ABB test sentences (47% ABA).**

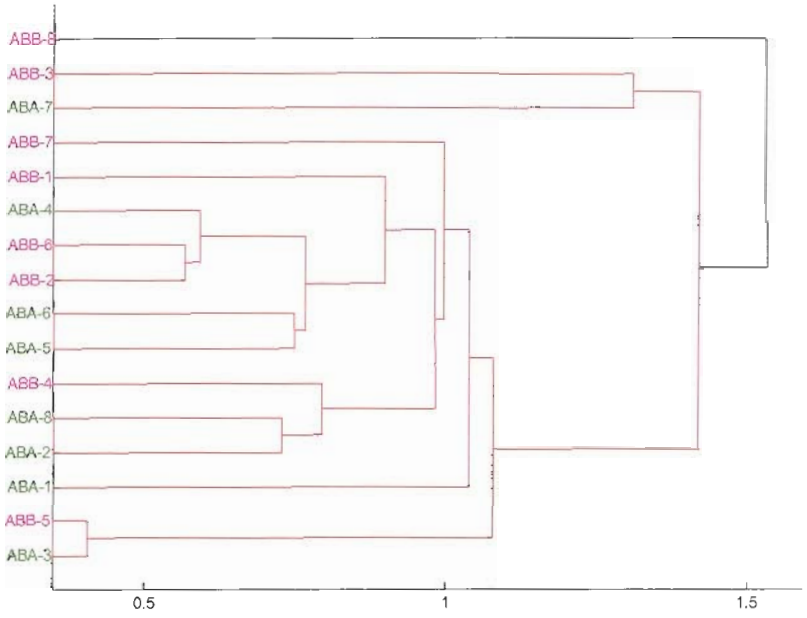

Figure 38 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #4 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 contains only 1 ABB sentence (100% ABB), whereas cluster 2 (red) contains 8 ABA and 7 ABB test sentences (47% ABB).**

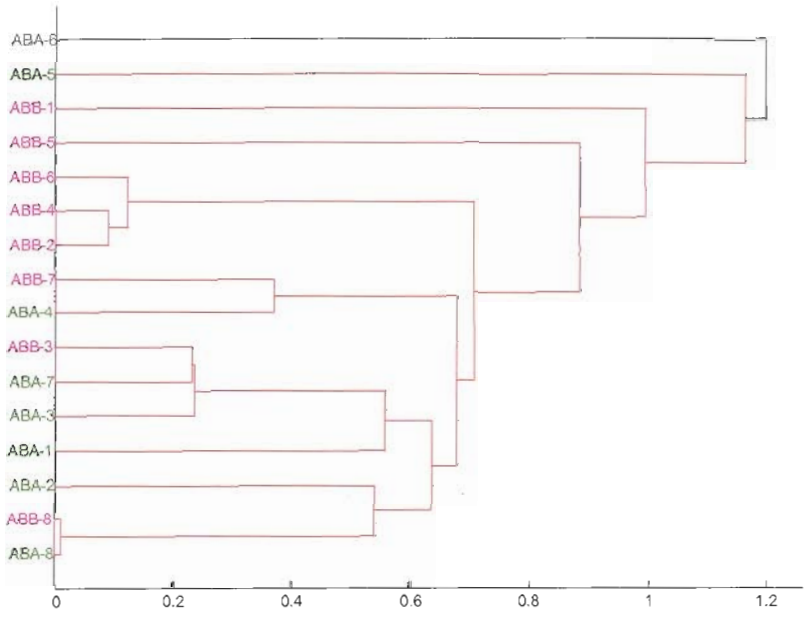

Figure 39 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #5 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 contains only 1 ABA sentence (100% ABA), whereas cluster 2 (red) contains 7 ABA and 8 ABB test sentences (47% ABA).**

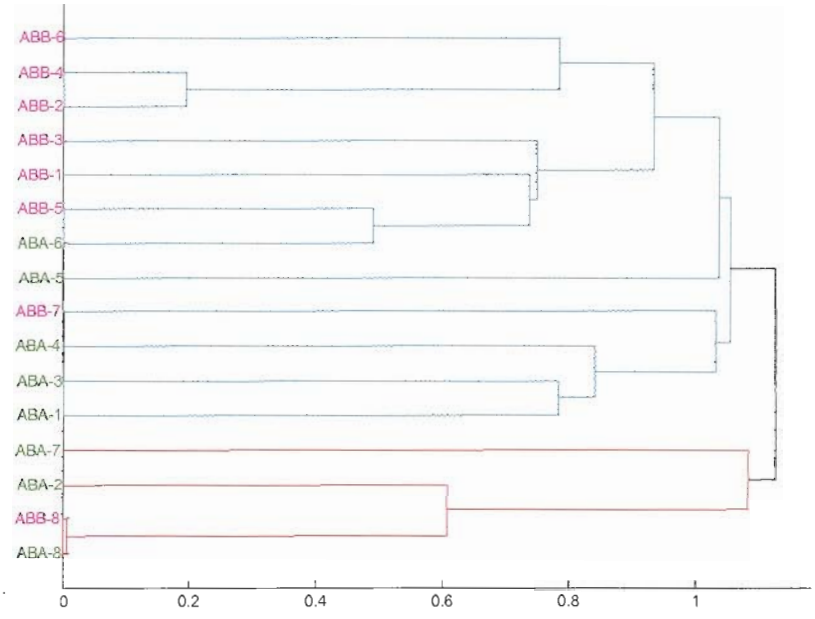

Figure 40 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #6 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 5 ABA and 7 ABB sentences (42% ABA), whereas cluster 2 (red) contains 3 ABA and 1 ABB test sentence (75% ABA).**

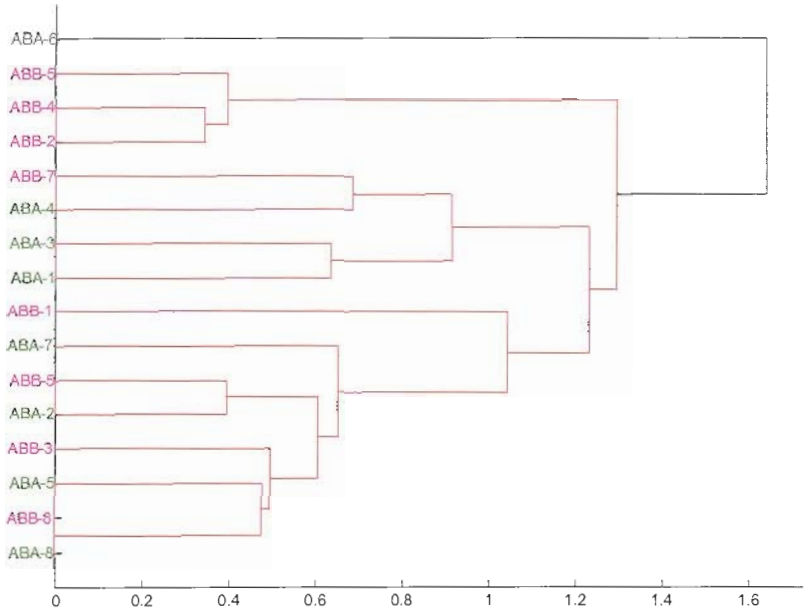

Figure 41 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #7 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 contains only 1 ABA sentence (100% ABA), whereas cluster 2 (red) contains 7 ABA and 8 ABB test sentences (47% ABA).**

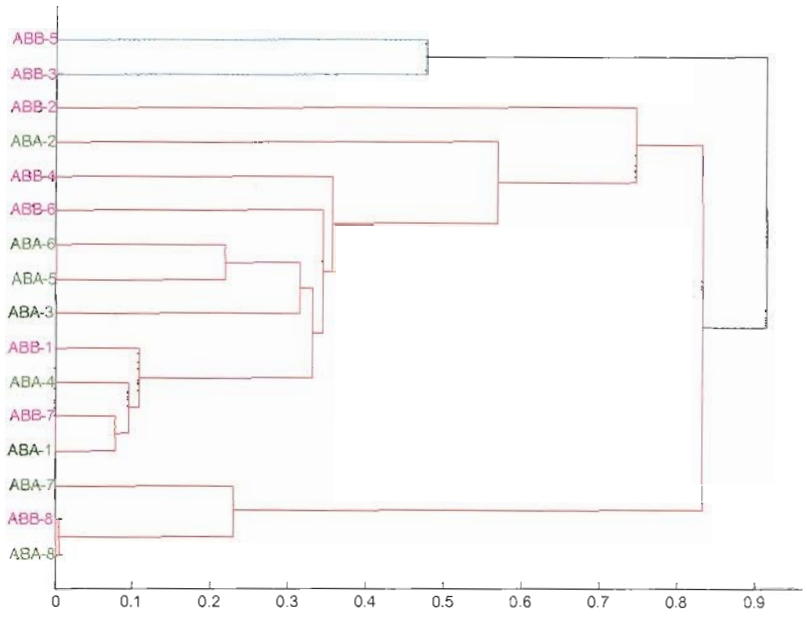

Figure 42 **The results of the cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #8 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 2 ABB sentences (100% ABB), whereas cluster 2 (red) contains 8 ABA and 6 ABB test sentence (43% ABB).**

5.7.2 Altmann's model

5.7.2.1 Principal component analysis

In the case of Altmann's (2002) model, PCA of hidden activations reveals fewer principal components than in Altmann & Dienes' (1999) case, reflecting a lesser degree of variation in the internal representations. The first 4 principal components account for about 95% of the variation. The projections of hidden activations, at the end of each sentence, onto the four principal components are displayed in Figures 43 and 44. The internal representations form several clusters depending on the type of "A" words that

appear at the end of each sentence (considering ABA training). Although some of these clusters are not completely disjoint, they are better formed than in the case of Altmann & Dienes (see Figures 43 and 44).

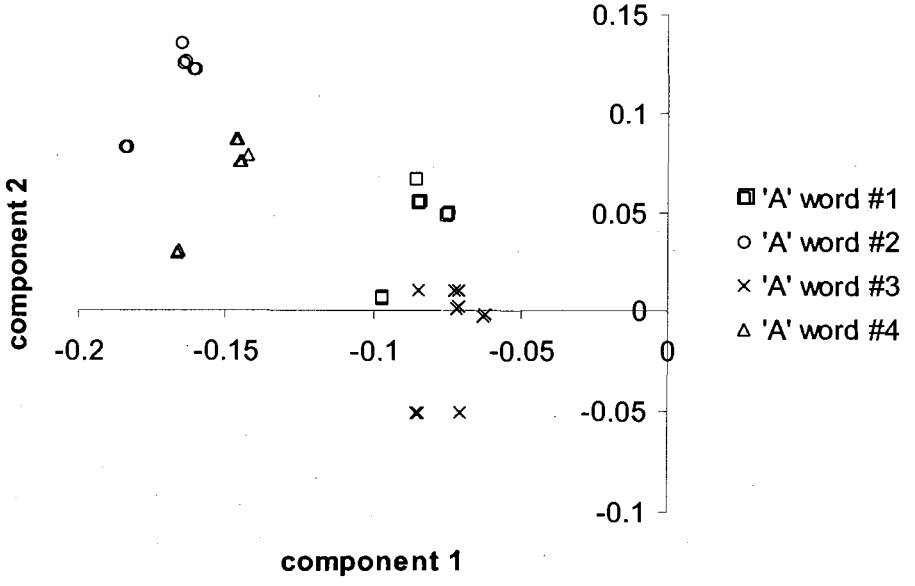

Figure 43 **Projections of hidden activations onto the first two principal components, at the end of each sentence (with regard to the "A" words that appear in the training sentences). There are clearly demarcated clusters among the internal representations formed by input sentences that end in each of the four "A" words.**

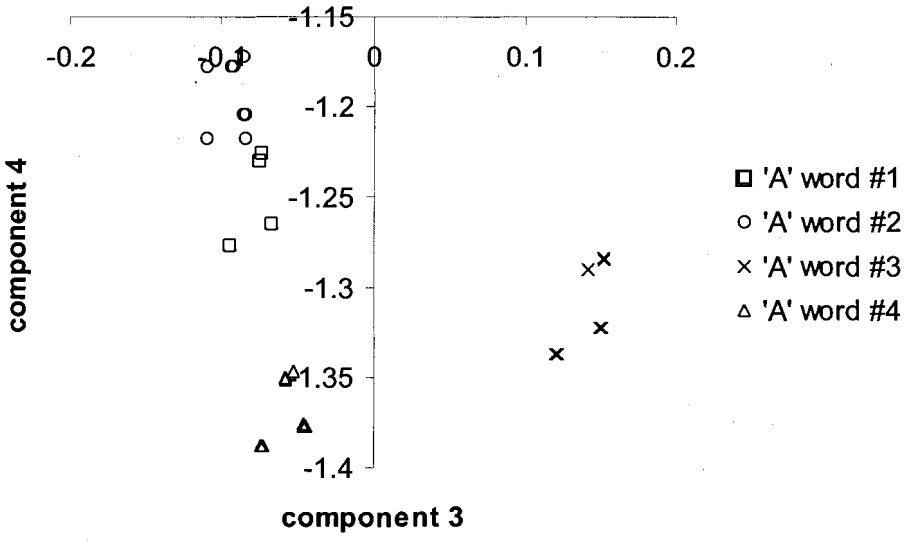

Figure 44 Projections of hidden activations onto the third and fourth principal components, at the end of each sentence (with regard to the "A" words that appear in the training sentences). There are clearly demarcated clusters among the internal representations formed by input sentences that end in each of the four "A" words.

In addition, Figure 45 displays the projections of hidden activations for *each word* of the training sentences onto the first two principal components. According to this graphic, very importantly, the network is able to detect the fact that the first and third words of each sentence (for ABA training) are similar, whereas the middle word is different. In part, this can originate from the fact that the input units associated to the "A" words are somewhat separated from the "B" units (a wider gap between units than in the Altmann & Dienes model). In any case, this may explain why this model has better results than Altmann & Dienes'.

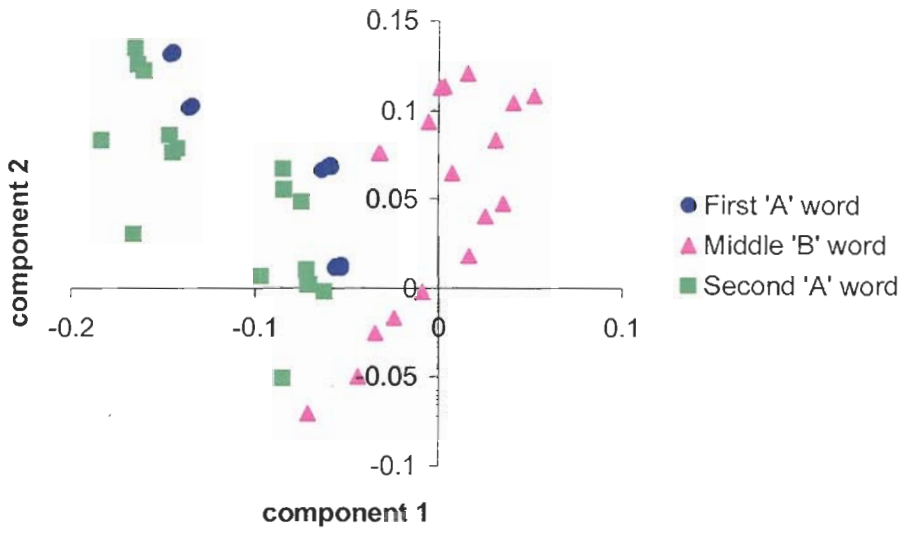

Figure 45 **Projections of hidden activations onto the first two principal components, for each word of the training sentences (with regard to the first "A" word, the middle "B" word, and the second "A" word). The internal representations formed by the "A" words are close to each other, and are fairly separated from the internal representations formed by the "B" words, indicating that the model is able to detect the fact that the first and last words are the same, whereas the middle word is always different.**

If there is a certain tendency of Altmann's model to form some meaningful internal representations when dealing with simple grammars (ABA vs. ABB), this tendency is completely lost when the model is presented with more complex (Gomez & Gerken) grammars. When simulating the fourth experiment of Gomez & Gerken (1999) on Altmann's model, the projection of hidden activations, at the end of each training sentence, onto the first two principal components, reveals an erratic and arbitrary behaviour of the network, the internal representations forming no apparent coherent structures (see Figure 46).

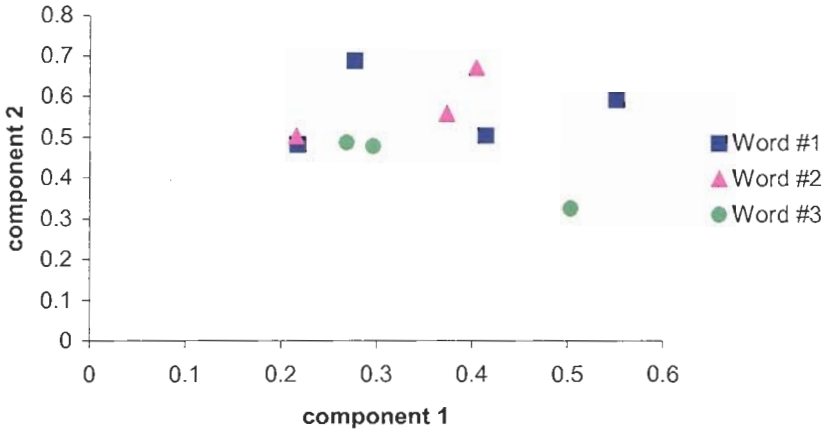

Figure 46 **Projections of hidden activations onto the first two principal components (with regard to the three words that can appear at the end of each training sentence) for a network trained with a finite-state grammar. There is no apparent grouping of the internal representations based on the words that can appear at the end of training sentences.**

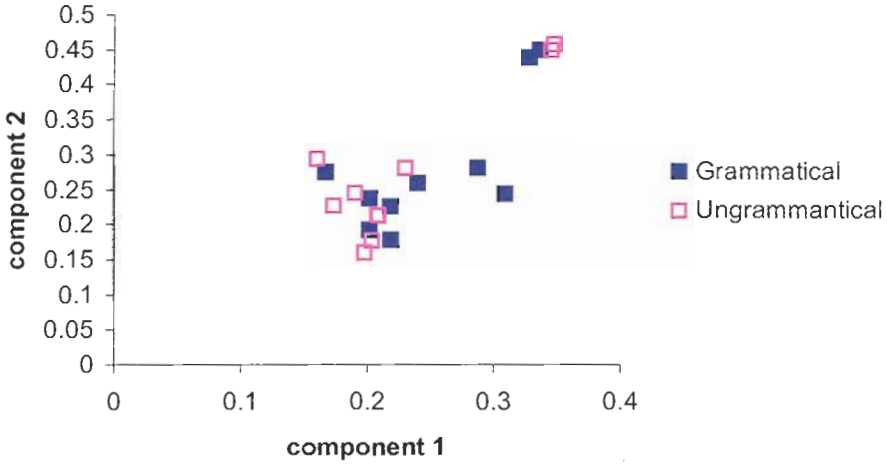

Figure 47 **Projections of hidden activations onto the first two principal components (with regard to the grammatical vs. non-grammatical patterns, at the end of each test sentence) for a network trained with a finite-state grammar. There is no apparent separation between internal representations of grammatical and ungrammatical test sentences.**

Also, Figure 47 displays the projections of hidden activations onto the first two principal components, at the end of each *test* sentence, in each condition (grammatical and ungrammatical). There is no apparent separation between the internal representations formed by the two categories of sentences, and this may explain why the network fails to distinguish between the two types of patterns.

5.7.2.2 Hierarchical cluster analysis

In order to see how the model groups the test sentences, a cluster analysis on the hidden activations at the end of each test sentence was performed. Because, originally, there were only 2 grammatical and 2 ungrammatical test sentences, to increase the statistical significance of the cluster analysis, 6 new test sentences in each condition were generated (the new sentences were generated by permuting the words in the original test sentences in a similar manner as described in Altmann & Dienes' case). Figures 48-55 display the results of the cluster analysis performed on 16 hidden activations (8 grammatical and 8 ungrammatical) for 8 different ABA-trained networks. Since I am interested in seeing whether the networks are able to separate the test stimuli into two groups, Figures 48-55 also display, in two different colours, how each of the 8 networks forms the two clusters and how they assign the hidden activation to the two clusters. Although there is a slightly lower degree of overlap (see Figures 48-55 for details) between grammatical and ungrammatical categories than in the case of Altmann & Dienes (1999), none of the 8 networks correctly separates all 16-test sentences into two groups.

The following 8 graphs (Figures 48 to 55) display the results of cluster analysis performed on 8 different networks (denoted network #1 to network #8).

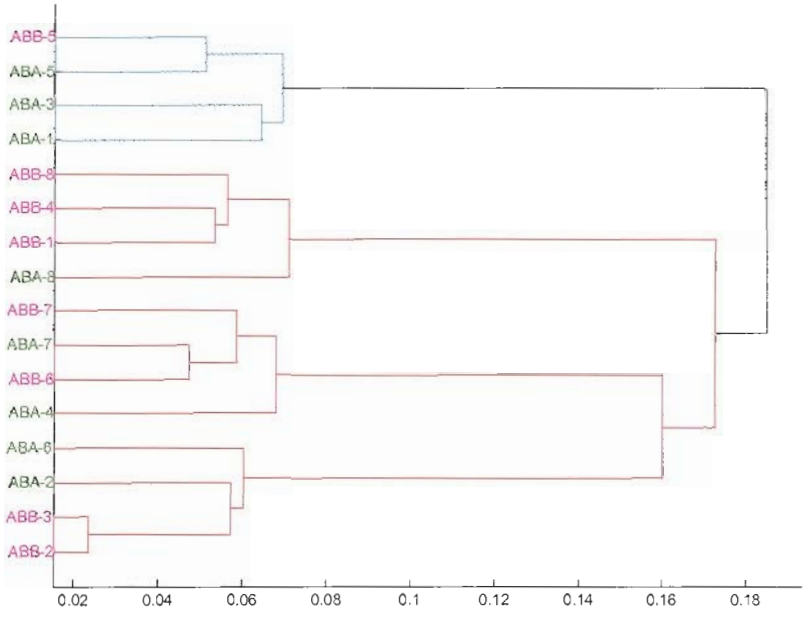

Figure 48 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #1 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABA and 1 ABB test sentence (75% ABA), whereas cluster 2 (red) contains 5 ABA and 7 ABB test sentences (42% ABA).**

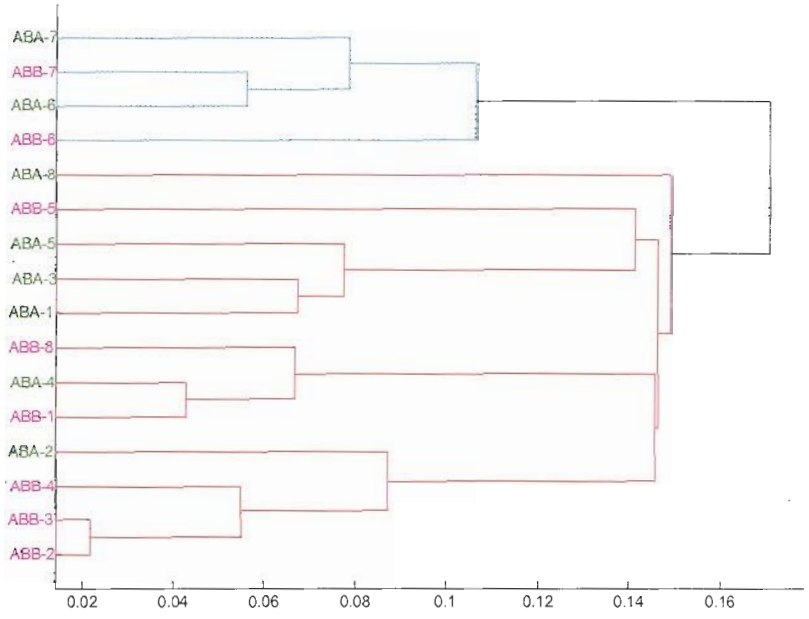

Figure 49 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #2 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 2 ABA and 2 ABB test sentences (50% ABA), whereas cluster 2 (red) contains 6 ABA and 6 ABB test sentences (50% ABA).**

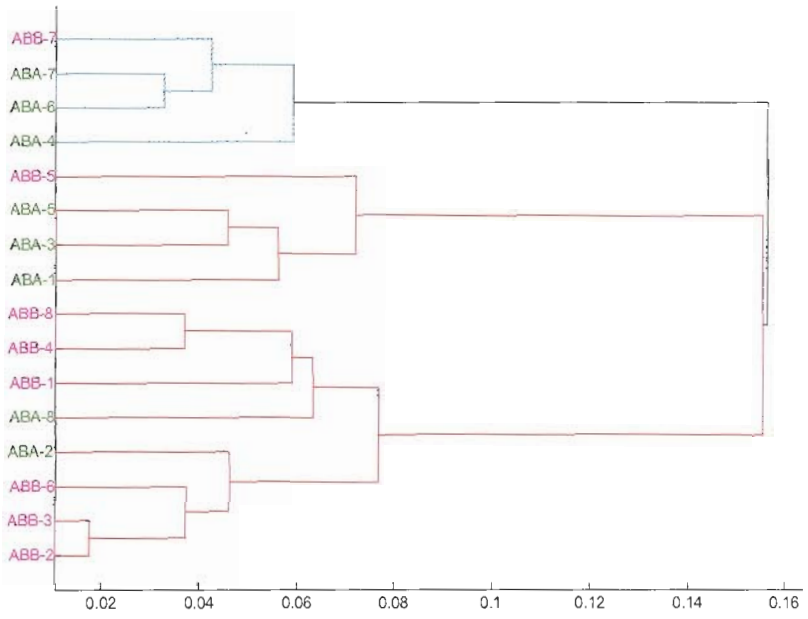

Figure 50 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #3 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABA and 1 ABB test sentence (75% ABA), whereas cluster 2 (red) contains 5 ABA and 7 ABB test sentences (42% ABA).**

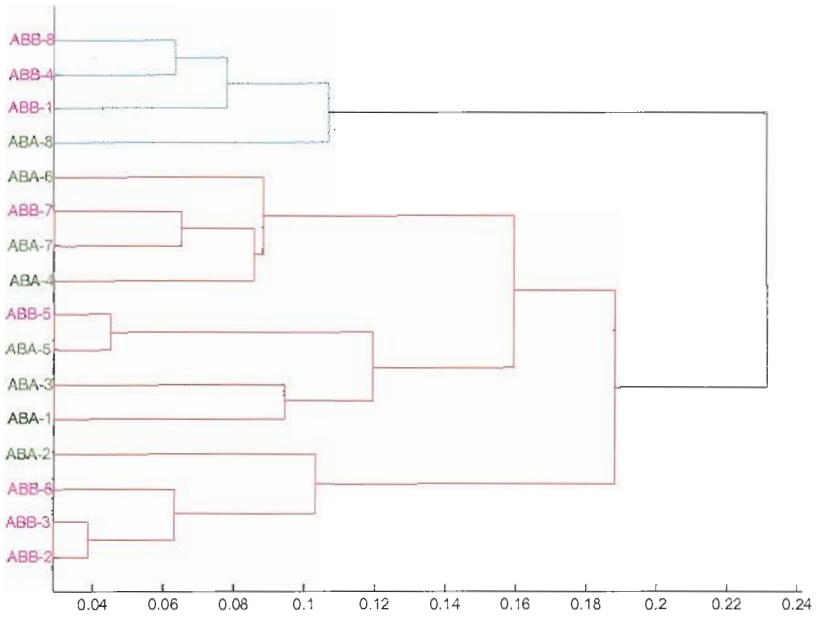

Figure 51 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #4 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABB and 1 ABA test sentence (75% ABB), whereas cluster 2 (red) contains 5 ABB and 7 ABA test sentences (42% ABB).**

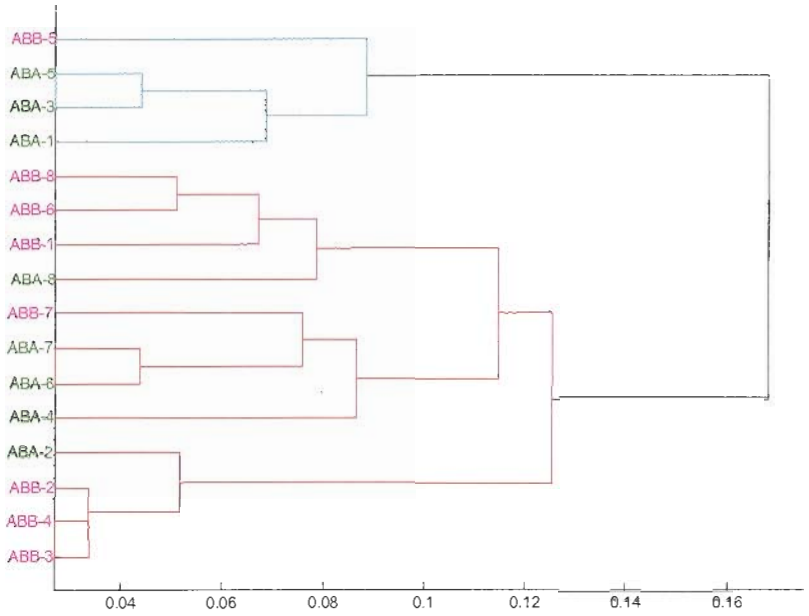

Figure 52 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #5 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABA and 1 ABB test sentence (75% ABA), whereas cluster 2 (red) contains 5 ABA and 7 ABB test sentences (42% ABA).**

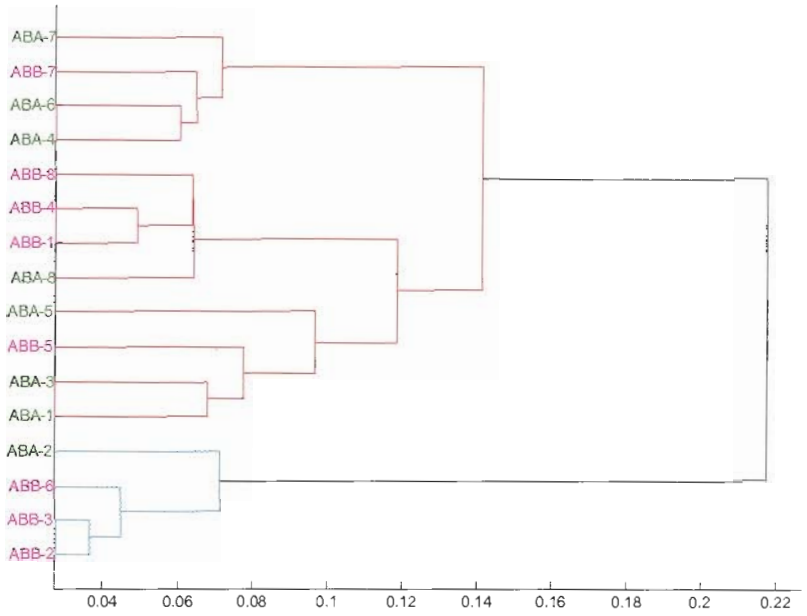

Figure 53 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #6 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABB and 1 ABA test sentence (75% ABB), whereas cluster 2 (red) contains 5 ABB and 7 ABA test sentences (42% ABB).**

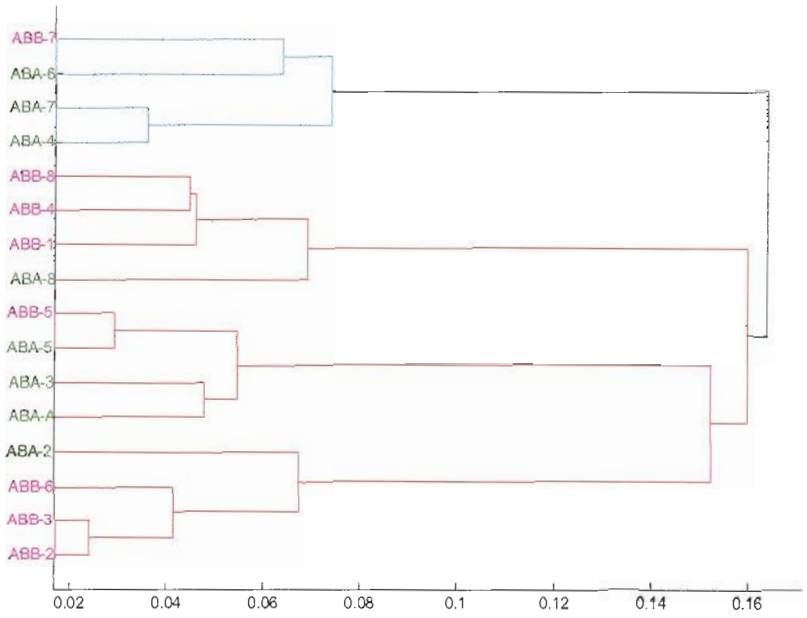

Figure 54 **The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #7 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABA and 1 ABB test sentence (75% ABA), whereas cluster 2 (red) contains 5 ABA and 7 ABB test sentences (42% ABA).**

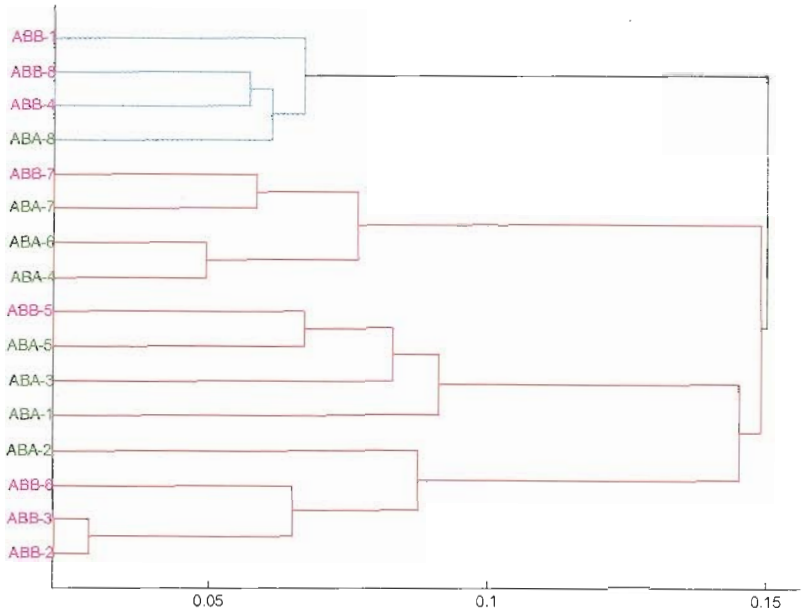

Figure 55 The results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 ABA and 8 ABB test sentences for network #8 (the horizontal values indicate the Euclidian distance between vectors). Cluster 1 (blue) contains 3 ABB and 1 ABA test sentence (75% ABB), whereas cluster 2 (red) contains 5 ABB and 7 ABA test sentences (42% ABB).

In order to see how the model groups the Gomez & Gerken test stimuli, a cluster analysis has been performed on 20 hidden activations formed at the end of 10 grammatical and 10 ungrammatical test sentences for one network trained with the Gomez & Gerken (1999) sentences (all other networks trained in this condition have the same behaviour). I am mostly interested in finding whether the network is able to cluster the data into two groups: grammatical and ungrammatical. Figure 56 displays the results of the cluster analysis, along with information, in two different colours, of how the network forms the two clusters and how it assigns the hidden activation vectors to either of the two clusters. According to Figure 56, there is a high degree of overlap between

hidden activations formed by grammatical sentences, and activations formed by ungrammatical sentences. This suggests that the model cannot robustly differentiate between the two categories of patterns.

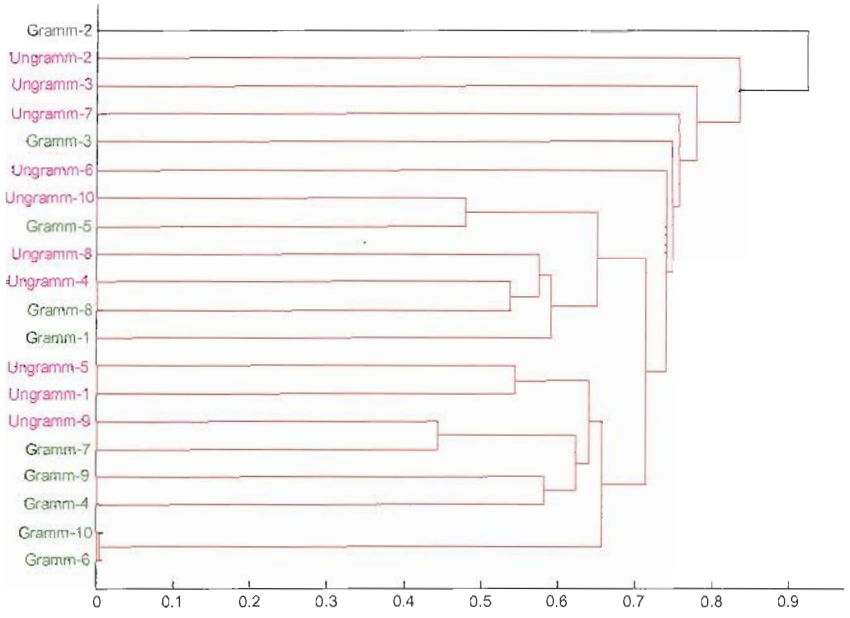

Figure 56 Results of cluster analysis on 20 hidden activation vectors formed at the end of 10 grammatical (Gramm-1 to Gramm-10) and 10 ungrammatical (Ungramm-1 to Ungramm-10) test sentences (the values on the horizontal line represent the Euclidian distances between vectors). Cluster 1 only contains 1 grammatical test sentence (100% grammatical), whereas cluster 2 (red) contains 9 grammatical and 10 ungrammatical test sentences (47% grammatical).

The way the model clusters the internal representations formed by the Gomez & Gerken test sentences indicates a high degree of overlap that exists among hidden activations. For instance, although the model groups the internal representations of Gramm-6 and Gramm-10, and, respectively, Gramm-4 and Gramm-9, these two grammatical subgroups do not immediately cluster with each other. Instead, they both

group with internal representations formed by ungrammatical sentences Ungramm-1, Ungramm-5, and Ungramm-9. The high degree the overlap visible in Figure 56 suggests that the model is unable to robustly separate the two categories of grammatical structures.

6 CHRISTIANSEN'S MODEL

6.1 Network Architecture and Preliminaries

To simulate Marcus *et al.*'s (1999) experiments, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) employ a simple recurrent network (see Figure 57), initially built to segment speech stream (Christiansen *et al.*, 1998). Christiansen & Curtin (1999), and Christiansen *et al.* (2001) describe the same model, the only difference between them being a small change in the input representation (see below). Both studies focus on the third experiment of Marcus *et al.*, where both grammars involved contain reduplications (ABB vs. AAB), and where there are no differences in phonetic features between the training and test syllables. Because of these characteristics, Marcus *et al.* (1999) argue that this third experiment cannot be driven by statistical mechanisms alone, and that infants must implement algebraic rules in order to perform the grammar discrimination task. Christiansen & Curtin (1999), and Christiansen *et al.* (2001) want to prove that even in this "difficult" case the input patterns carry sufficient statistical information that can allow the networks to successfully replicate Marcus *et al.*'s results without implementing algebraic rules. Christiansen *et al.* assume that, if this is true for the third experiment, the first two experiments of Marcus *et al.* will be much easier to replicate by their model.

6.2 Input Representations

At input, Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001) networks are provided with three bits of information (cues):

- 18 phonological features on the input[30], and 36 phonemes on the output layer.
- Utterance (sentence) boundary (U-B) information marking sentence endings (the units marked "#" in Figure 57).
- Lexical stress: no stress, secondary stress (unit S), and primary stress (unit P).

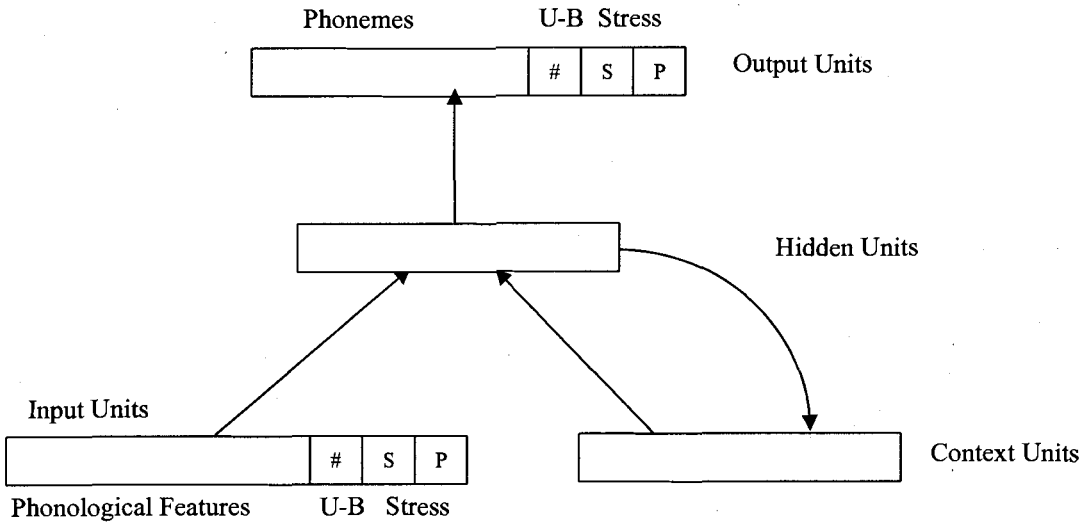

Figure 57 Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001) model

The sentences are presented to the network as sequences of words, each word being separated into phonemes. The phonemes are presented one at a time, each phoneme being represented at input by a set of phonological features, along with information about sentence boundaries (U-B unit) and stress (primary stress, secondary stress, or no stress). Each bit in the input vector corresponds to a phonological feature; if a phoneme has a certain feature, the corresponding bit will be set to 1, otherwise the bit will be 0. The

[30] 18 phonological features are used by Christiansen *et al.* (2001). However, Christiansen & Curtin (1999) use only 11 features (similar to the original Christiansen *et al.*, 1998).

output representation is localist: the output layer consists of 36 units, one unit for each phoneme[31], plus one utterance-boundary (U-B) unit, and two units to indicate stress. The U-B units are activated when the last phoneme of the last word of a sentence is presented. The words are transformed from orthographic format to phonological form (including lexical stress) using the MRC Psycholinguistic Database from the Oxford Text Archive. Some of the phonological features used in this representation are: sonorant, consonantal, voice, nasal, degree, palatal, pharyngeal, etc. For instance, the phoneme *n* from the word *no* has *sonorant*=0, *voice*=1, *nasal*=1, *pharyngeal*=0, etc. For all monosyllabic words, the stress is always primary, i.e., unit P is set to 1 and unit S is set to 0. For words such as *beautiful*, the stress is on the first syllable (primary stress), while the other two syllables do not have stress (both stress units are set to 0).

6.3 Training and Testing

The network is trained to predict the next phoneme in a word, as well as the values for stress and utterance (sentence) boundary. Christiansen & Curtin (1999), and Christiansen *et al.* (2001) believe that the network is able to integrate the input cues, and at the end of training it is capable of predicting not only the sentence boundaries, but also the word boundaries. Christiansen *et al.* (2001) use 16 different networks (each network resembles one infant in Marcus *et al.*'s simulation), having different sets of initial weights, randomized in the interval [-0.25, +0.25]. The learning rate was set to 0.1, while the momentum was 0.95.

[31] At the input layer, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) use the phonological representation of phonemes (distributed representation), while at the output layer, they only activate one unit per phoneme (localist representation), in order to facilitate an easier analysis of the network's results.

Similar to Elman's (Elman, 1999; Seidenberg & Elman, 1999) and Altmann's (Altmann, 2002) simulations, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) precede the training and test phases with a preliminary training phase, during which a multitude of speech utterances (sentences) is presented to the network in a single pass. In this phase, the network is trained to segment words spoken to infants aged 6-16 weeks. The pre-training corpus is part of CHILDES database (MacWhinney, 1991)[32]. The corpus has a total of 24,648 words, with an average sentence length of 3 words. Like Elman, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) argue that the reason for this pre-training phase is to simulate the fact that "the 7-month-olds in the Marcus *et al.* study already have had a considerable exposure to language, and have begun to develop their speech segmentation abilities" (Christiansen *et al.*, 2001).

After pre-training, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) train and test the model with sentences used in the Marcus *et al.*'s third experiment: 16 AAB and 16 ABB training sentences, and 2 AAB and 2 ABB test sentences. For each phoneme in the training and test corpora that is presented to the networks, they record the activation of the utterance boundary output unit, and calculate the average of boundary unit activations for all training phonemes. Whenever the activation of the output boundary unit for a test phoneme is higher than this average, it is considered that an end of word is detected.

[32] Examples of pre-training sentences: *Are you a sleepy head?, What a milky face!,* etc.

6.4 Results

Christiansen & Curtin (1999), and Christiansen *et al.* (2001) measure the performance of their model by computing the completeness scores for consistent (grammatical) and inconsistent (ungrammatical) sentences:

Completeness = Hits / (Hits + Misses)

This is a measure of how many words the network is able to discover, i.e., how well the consistent and inconsistent sentences are segmented.

For instance, if two sentences were *ba ba po*, and *ko ga ga*, and the network segmented them as *bab#a#po#ko#gag#a* (# represents the predicted word boundary), it would discover two words *po* and *ko* (Hits = 2), and miss 4 words. This means a completeness score of 2 / (2 + 4) = 33.3%.

Christiansen & Curtin (1999), and Christiansen *et al.* (2001) found that there is a difference in the way their model segments the words in sentences generated with the familiar (consistent sentences) and unfamiliar grammars (inconsistent sentences): the average completeness score for consistent sentences is 28.82%, while the average score for inconsistent sentences is 35.76%. According to Christiansen & Curtin (1999), and Christiansen *et al.* (2001), this means that the networks are better at segmenting words in inconsistent sentences than in consistent ones. Their conclusion is that the "knowledge acquired in the service of learning to segment the speech stream can be recruited to carry out the kind of classification task used in the experiment by Marcus *et al.*" (Christiansen & Curtin, 1999), and therefore statistical mechanisms can indeed account for the infants' results.

6.5 Marcus' Evaluation

Marcus has, however, a few concerns about this model being capable of capturing his results:

- Marcus has the impression that Christiansen & Curtin (1999) implicitly suggest that infants can discern word boundaries in test sentences but not in the habituation sentences, which, he believes, "makes little sense" (Marcus, 2001). Indeed, Christiansen & Curtin (1999) claim that their model produces better results when segmenting consistent sentences than inconsistent sentences, and this is an analogy to infants spending more time listening to inconsistent items. Although this analogy is not clearly described (Christiansen & Curtin do not explain why segmentation ability should correspond to grammaticality, and, more importantly, why their model performed better when presented with inconsistent items), it is true that infants can display both familiarity-preference, and novelty-preference, depending on how the experiment is conducted. Christiansen & Curtin apparently show "a differential ability to predict word boundaries for the words in the two test conditions" (Christiansen & Curtin, 1999), but their attempt to explain this in terms of "inconsistent items [that] *stand out* more clearly in comparison with the consistent items" (Christiansen & Curtin, 1999) (my emphasis) is, at best, unfortunate. I believe that Christiansen & Curtin's highly psychological language used to explain the network's functional behaviour is very inappropriate.

- Marcus argues that Christiansen & Curtin's results may be caused by noise, and that these results are not statistically significant. Christiansen & Curtin's (1999) initial study was performed on only one network, and their result can easily be viewed as being accidental. However, in their more recent simulation, Christiansen *et al.* (2001) claim that

they replicated their initial result on 16 different networks. If this were true, it would make Marcus' argument less powerful.

6.6 Personal Investigation

During my own analysis of this model, I discovered that Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001) claims are mostly overstated. I repeated their experiment on 32 different simple recurrent networks, using the same training parameters, input corpora, and evaluation criterion, with two different simulators (VNNS and PDP++). My results are not as good as Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001). First of all, they report completeness scores in the range of 30%, whereas the completeness scores computed on the networks that I trained are less than 10% (in general, higher completeness scores were measured for unfamiliar test sentences than for familiar ones). Secondly, a statistical check on these results show that they are not significant across the 32 networks that I trained. An analysis of variance (ANOVA) measurement generated $F = 3.99$, and $p=0.35$. This means that the networks contain significant noise, which supports Marcus' account on this model. Additionally, I trained the model on more complex grammars (such as the Gomez & Gerken's grammars shown in Figures 5 and 6) using 32 different networks (each network had its own set of initial weights). None of these networks generated results that were consistent with Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001) claims (ANOVA: $F=4.12$, $p=0.32$).

In conclusion, Marcus' concerns about this model proved to be right. Although, unlike the other networks discussed in this paper, Christiansen & Curtin's (1999), and Christiansen *et al.*'s (2001) model was not specifically designed to simulate Marcus *et*

al.'s experiment[33], the authors claim they showed that "an existing connectionist model of early infant word segmentation (Christiansen *et al.*, 1998) could utilize statistical knowledge acquired in the service of speech segmentation to fit the infant data from Marcus et al. (1999; Experiment 3) under very naturalistic circumstances" (Christiansen *et al.*, 2001). However, I have serious concerns about the validity of this statement, and I agree with Marcus that the model is mostly driven by noise. My knowledge representation analysis of this model proved this.

6.7 Knowledge Representation Analysis

What is characteristic of this SRN model is the fact that, contrary to other models discussed here, Christiansen & Curtin (1999), and Christiansen *et al.* (2001) only perform the third experiment of Marcus *et al.* (1999), which involves the grammars AAB and ABB. Unless specified otherwise, all hidden activations used in this analysis have been collected at the end of each input sentence. But, since in this model the input words are not presented to the network as a whole, but as sequences of two phonemes, the hidden activations formed by both phonemes of the last word of each sentence have been used (both the vowel and consonant phonemes have been used in analysis).

6.7.1 Principal component analysis

PCA of hidden activations at the end of each sentence reveals four principal components (which together account for about 95% of the variation). The first principal component (~60% of the variation) separates all hidden activations into two well-formed clusters: activations generated by consonants and those generated by vowels. Figure 58

[33] Christiansen & Curtin (1999), and Christiansen *et al.* (2001) view this as a feature, not a disadvantage.

shows that, with regard to the first principal component, the internal representations formed by the vowels of the last word of each sentence are fairly separated from the representations formed by the consonants of those words. The next principal component (~20% of the variation) reveals the difference among various individual letters of the last words of each sentence: the activations generated by letters "d" and "l" are very close to each other (reflecting the fact that their input representations are very similar). At the same time, the activations for letters "w" and "j" are close to each other (because the difference between their input representations is only one bit). The activations formed by the letters "e" and "i" are quite separate from the activations formed by all other letters.

A projection of the *average* hidden activations[34] onto the third and fourth principal components (which account for 10% and 5% of the variation, respectively) reveals four clusters (see Figure 59). Each cluster corresponds to a word that appears at the end of each sentence: "di", "li", "we" and "je". The group of activations formed by the words "di" and "li", on one hand, and the group formed by the words "we" and "je", on the other hand, are very close to each other, reflecting the similarity between the input representations of letters "d" and "l", and "w" and "j", respectively (which is consistent with the previous projections shown in Figure 58).

[34] This is the average between the hidden activation vectors generated by the consonant (first phoneme) and the vowel (second phoneme) that form the last word of each sentence. I used the average because I wanted to capture the network's response to the whole word.

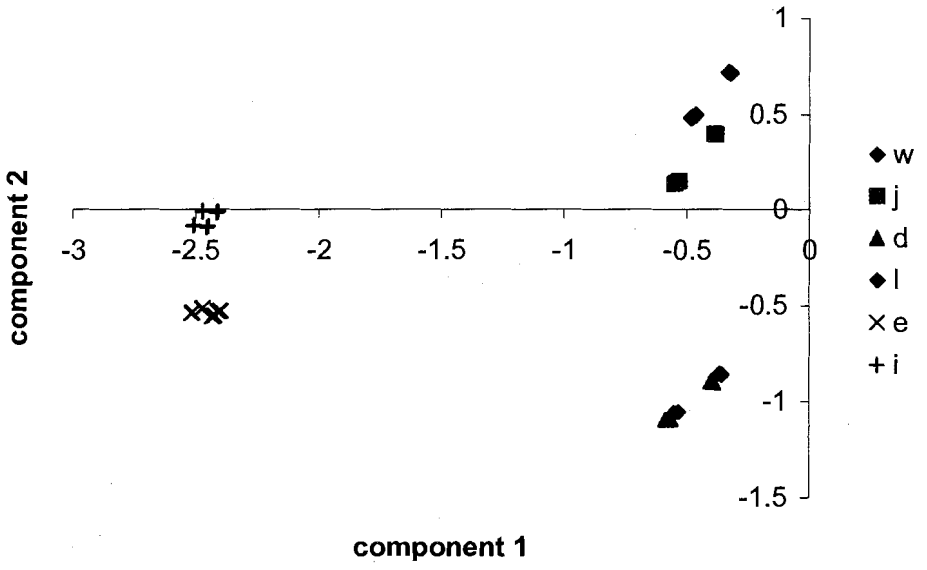

Figure 58 Projections of hidden activations onto the first two principal components (with regard to the letters that occur in the last word of each sentence). There is a clear separation along the first principal component between the internal representations formed by consonants and those formed by vowels. The second principal component reveals the differences among various input words.

PCA of hidden activations (Figures 58 and 59) reveals that most of the network resources (80%) are used to identify the various characteristics of the input words (such as the separation between words that end in "e" from those that end in "i"), information which is not at all useful to the more important task of learning the sequential structure of the input sentences. This finding is confirmed by the following two graphs, which demonstrate that the model is unable to abstract the sequential structure of the input stimuli.

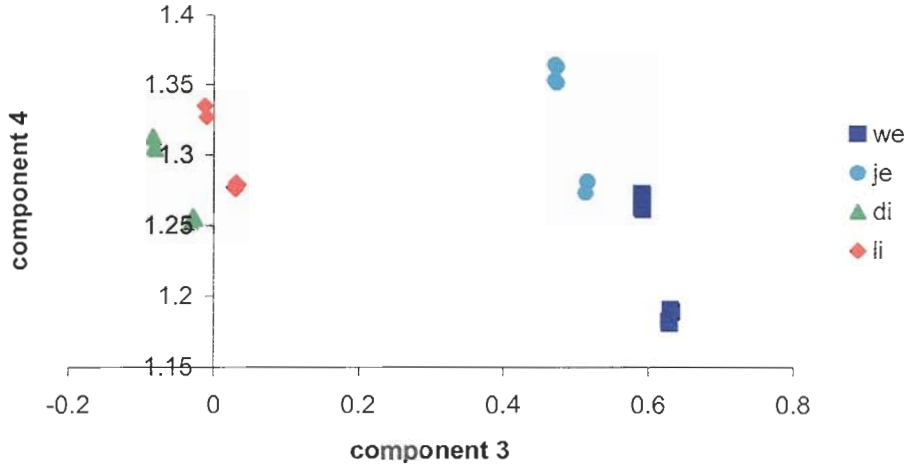

Figure 59 **Projections of average hidden activations onto the third and fourth principal components (with regard to the words that occur at the end of each sentence). The internal representations form several groups depending on the input words. The relative distances between groups indicate the differences between input representations of those words.**

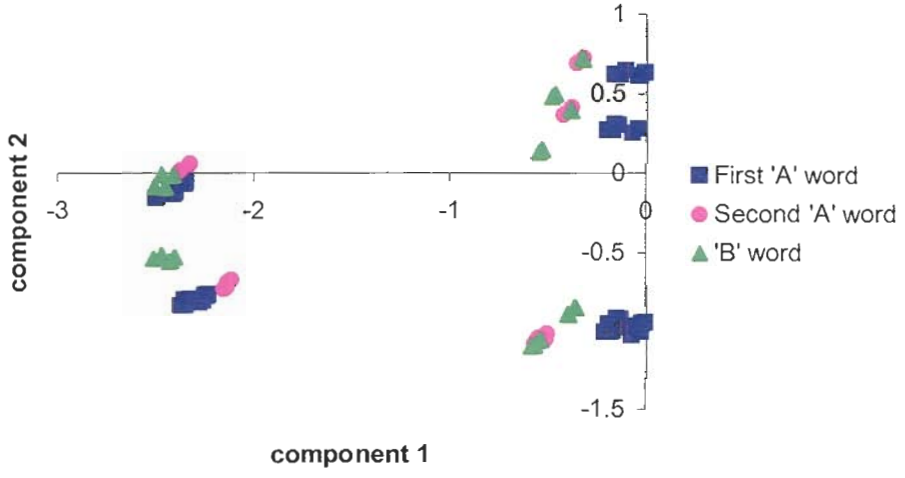

Figure 60 **Projections of all hidden activations onto the first two principal components (with regard to the first and second "A" words, and the "B" word in each AAB sentence). There are no apparent separations between input representations formed by the first and second "A" words, and those formed by the "B" words.**

Figures 60 and 61 display the projections of hidden activations onto the first four principal components of *each* input word within input sentences (during AAB training).

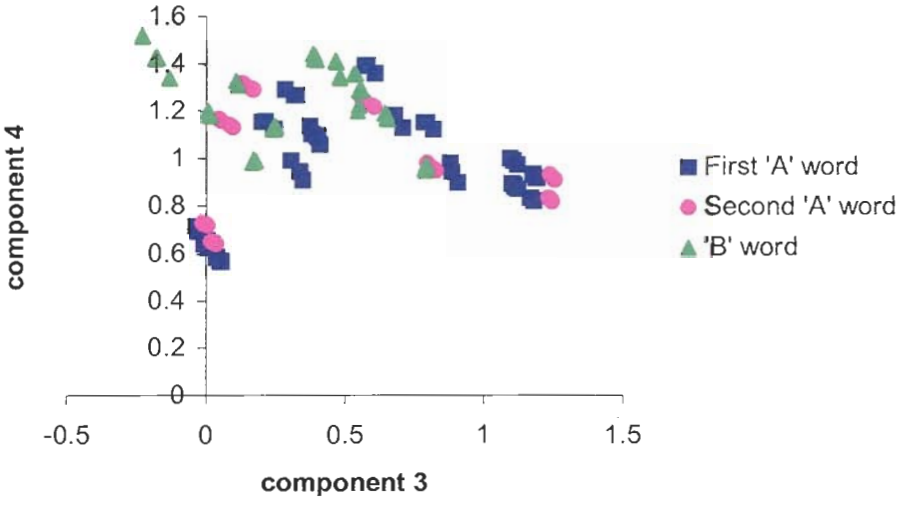

Figure 61 Projections of all hidden activations onto the third and fourth principal components (with regard to the first and second "A" words, and the "B" word in each AAB sentence). There are no apparent separations between input representations formed by the first and second "A" words, and those formed by the "B" words.

According to Figures 60 and 61, there is a large degree of overlap between internal representations formed by "A" and "B" words, indicating that the network may not differentiate the words that appear on the first and second positions (the "A" category for AAB training) from the words that appear on the third position (the "B" category), even though none of the "B" words is ever used as an "A" word. The network does not seem to be aware of any particular grammatical structure of the input patterns. Therefore, the fact that the network is presented with AAB or ABB test patterns does not influence the network's ability to form specific internal representations, which may explain the amount of noise found in my experimental results. PCA of hidden activations during the

test phase confirms this hypothesis: the network develops similar internal representations for both AAB and ABB test sentences. Figures 62 and 63 display the projections of hidden activations onto the first four principal components, for both grammatical and ungrammatical test sentences (the activations are recorded at the end of each sentence). According to these two pictures, there is an important degree of overlap between the internal representations formed by grammatical test sentences, and those formed by ungrammatical test sentences. This may explain why the network cannot robustly differentiate between the two categories of sentences.

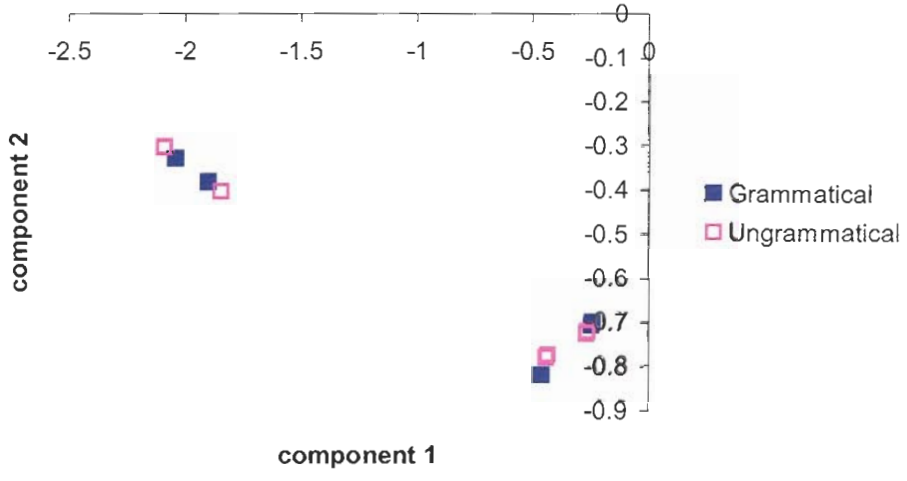

Figure 62 **Projections of hidden activations at the end of each test sentence onto the first two principal components (with regard to the grammatical and ungrammatical test sentences). There is a high degree of overlap between internal representations formed by grammatical and ungrammatical test sentences.**

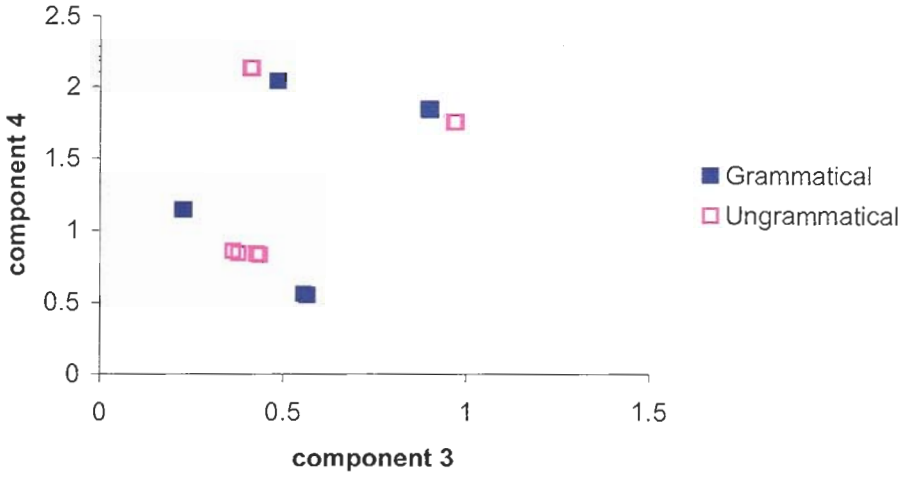

Figure 63 **Projections of hidden activations at the end of each test sentence onto the third and fourth principal components (with regard to the grammatical and ungrammatical test sentences). There is a high degree of overlap between internal representations formed by grammatical and ungrammatical test sentences.**

PCA of contributions (products between hidden activations and output weights) produces a similar result (the projections of network contributions are very similar to those shown in Figures 58-63).

Based on the analysis performed up to this point, the fact that this model (also) fails when using more complex grammars is expected. Indeed, when using the Gomez & Gerken's grammars, PCA of hidden activations and network contributions in this condition reveals a very similar picture: the network is able to identify fairly well the individual letters of the input set, but there is no information in the hidden activations or output weights about the categories of words or the syntactical structure of sentences that are presented to the network. Figure 64 displays the projections of network contributions onto the first two principal components, for both grammatical and ungrammatical test

sentences (the contributions are recorded at the end of each sentence). According to this graphic, the network contributions for grammatical and ungrammatical test sentences exhibit a very high degree of overlap. This may explain why the network is not able to robustly differentiate between grammatical and ungrammatical structures, as shown by my experimental results.

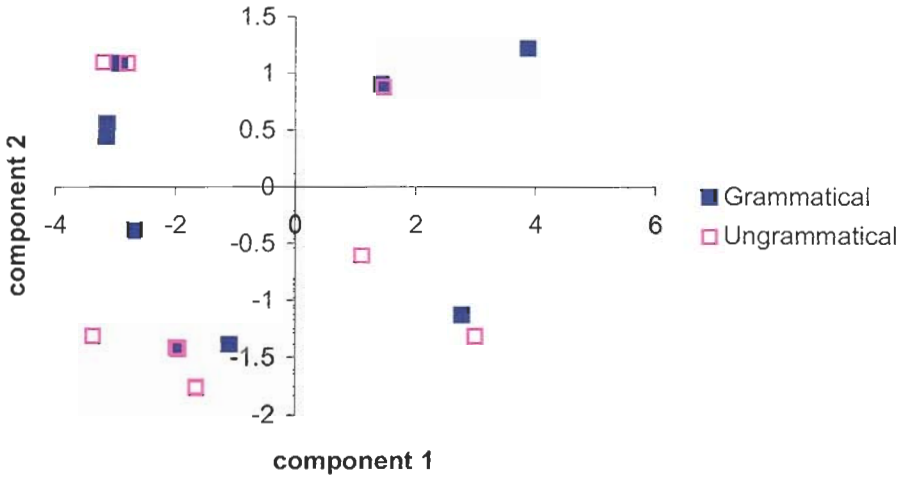

Figure 64 **Projections of network contributions onto the first two principal components (with regard to the grammatical and ungrammatical test sentences formed with Gomez & Gerken grammars). There is a high degree of overlap between the network contributions formed by grammatical test sentences and the network contributions formed by ungrammatical test sentences.**

6.7.2 Hierarchical cluster analysis

A cluster analysis performed on the hidden activations at the end of each test sentence also reveals that this model is unable to robustly and consistently differentiate between grammatical and ungrammatical test patterns. Because, originally, there are only 2 grammatical and 2 ungrammatical test sentences, to increase the statistical significance

of this analysis, 6 new test sentences, in both conditions, were generated (by permuting the words in the original test sentences, in a similar manner as in Altmann & Dienes' case). Figures 65-72 display the results of the cluster analysis performed on 16 hidden activations formed at the end of 8 grammatical and 8 ungrammatical test sentences, for 8 different AAB-trained networks (denoted network#1 to network#8). Since I am mostly interested in finding whether the networks are able to group the hidden activations into two clusters (grammatical vs. ungrammatical), Figures 65-72 also display, in two different colours, how the networks actually form the two clusters and how they assign each hidden activation vector to those clusters. According to Figures 65-72, none of the 8 networks are able to correctly categorize all 16 internal representations. There exists a high degree of overlap between activations formed by grammatical and ungrammatical test sentences (see Figures 65-72 for details regarding the overlap), confirming the previous finding that the networks cannot robustly differentiate between the two categories of sentences.

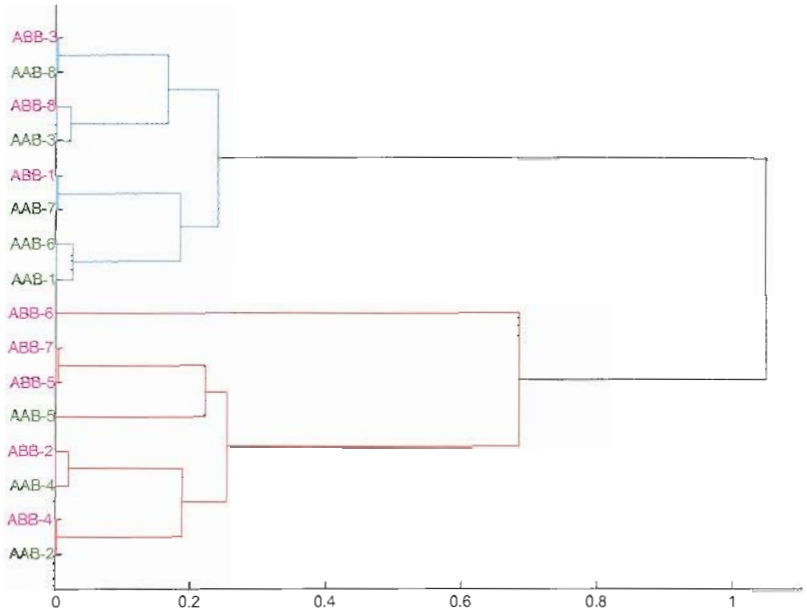

Figure 65 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #1 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 5 AAB and 3 ABB sentences (63% AAB), whereas cluster 2 (red) contains 3 AAB and 5 ABB sentences (38% AAB).**

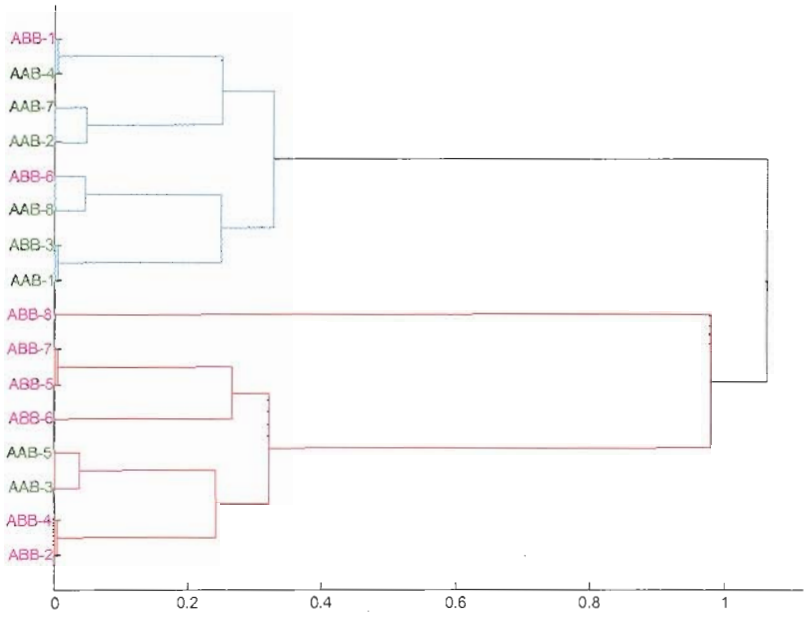

Figure 66 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #2 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 5 AAB and 3 ABB sentences (63% AAB), whereas cluster 2 (red) contains 3 AAB and 5 ABB sentences (38% AAB).**

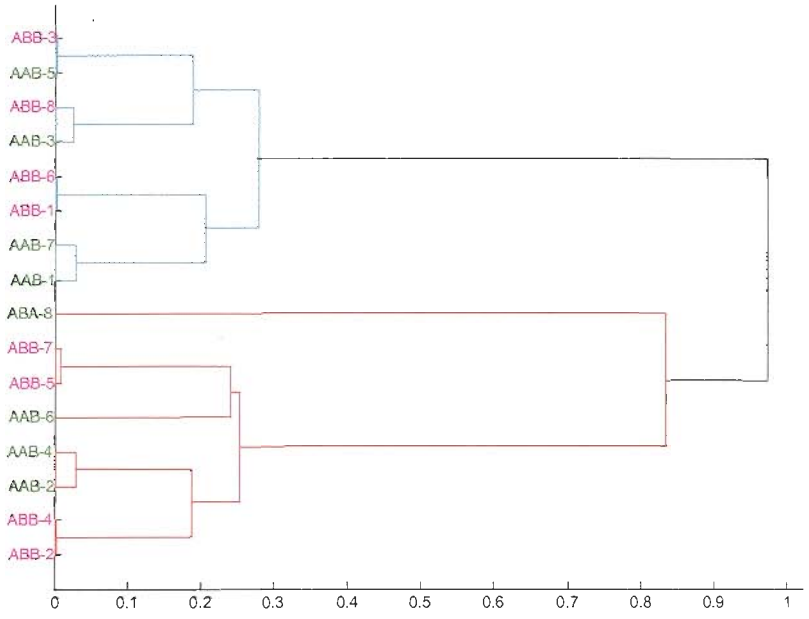

Figure 67 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #3 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 4 AAB and 4 ABB sentences (50% AAB), whereas cluster 2 (red) contains 4 AAB and 4 ABB sentences (50% AAB).**

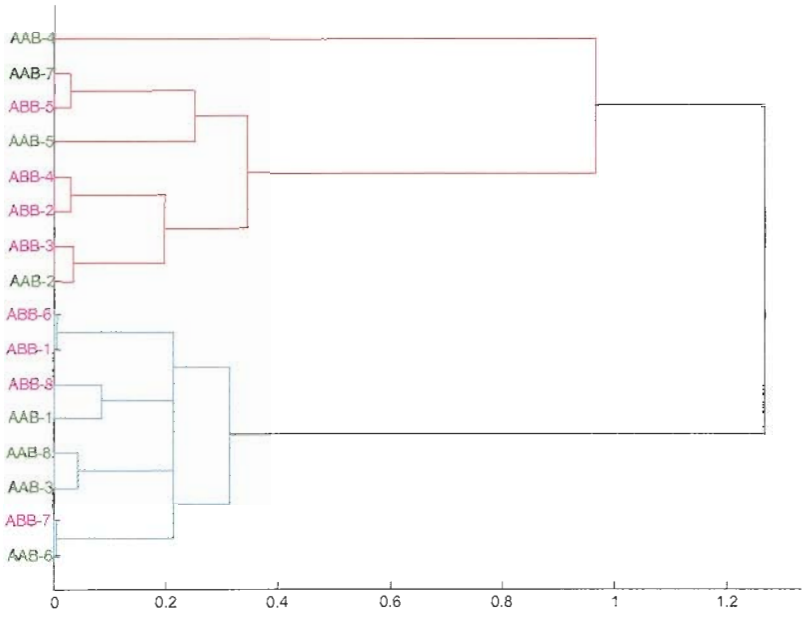

Figure 68 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #4 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 4 AAB and 4 ABB sentences (50% AAB), whereas cluster 2 (red) contains 4 AAB and 4 ABB sentences (50% AAB).**

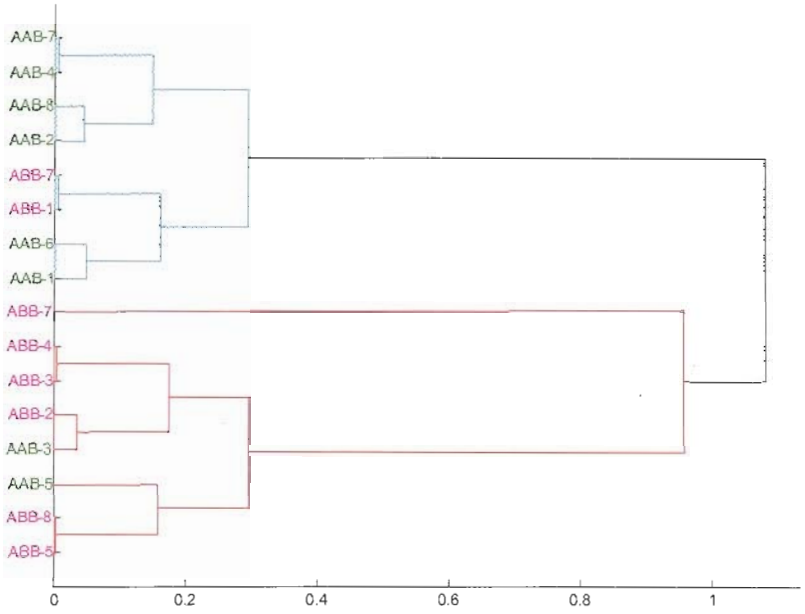

Figure 69 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #5 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 6 AAB and 2 ABB sentences (75% AAB), whereas cluster 2 (red) contains 2 AAB and 6 ABB sentences (25% AAB).**

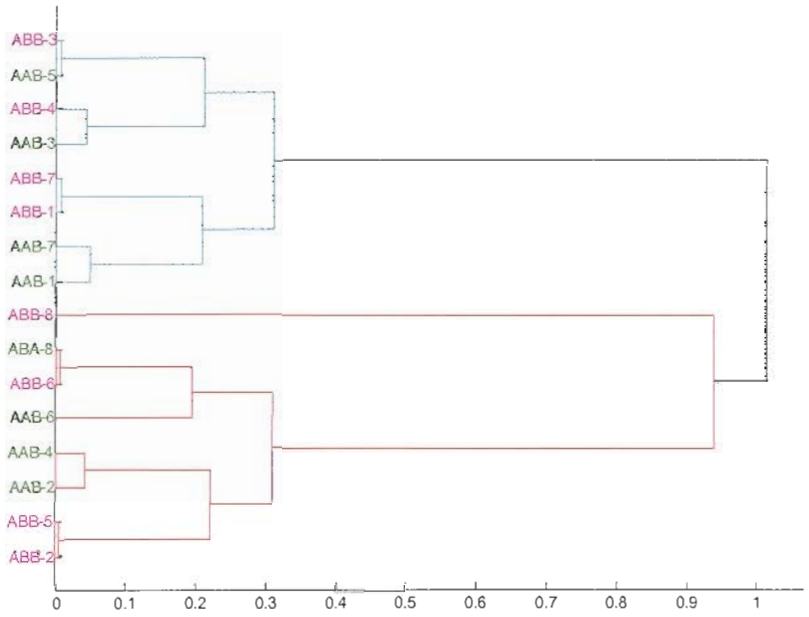

Figure 70 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #6 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 4 AAB and 4 ABB sentences (50% AAB), whereas cluster 2 (red) contains 4 AAB and 4 ABB sentences (50% AAB).**

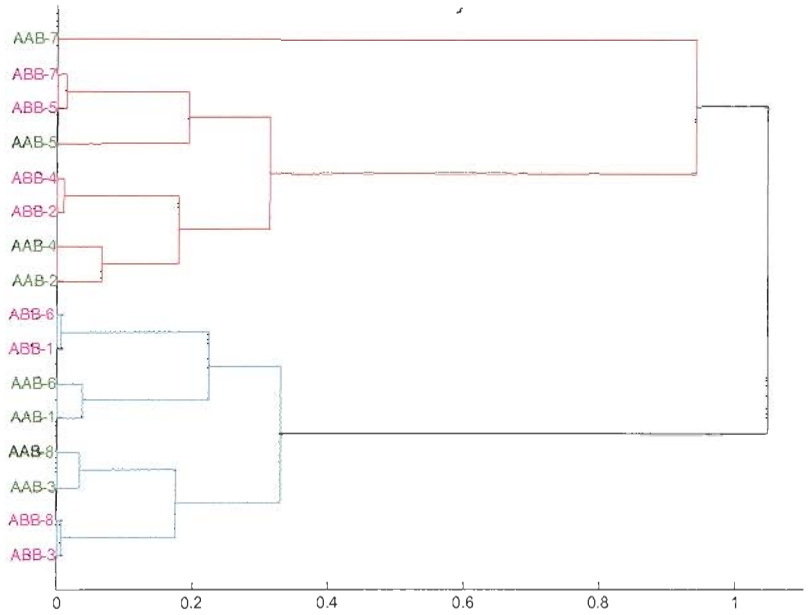

Figure 71 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #7 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 4 AAB and 4 ABB sentences (50% AAB), whereas cluster 2 (red) contains 4 AAB and 4 ABB sentences (50% AAB).**

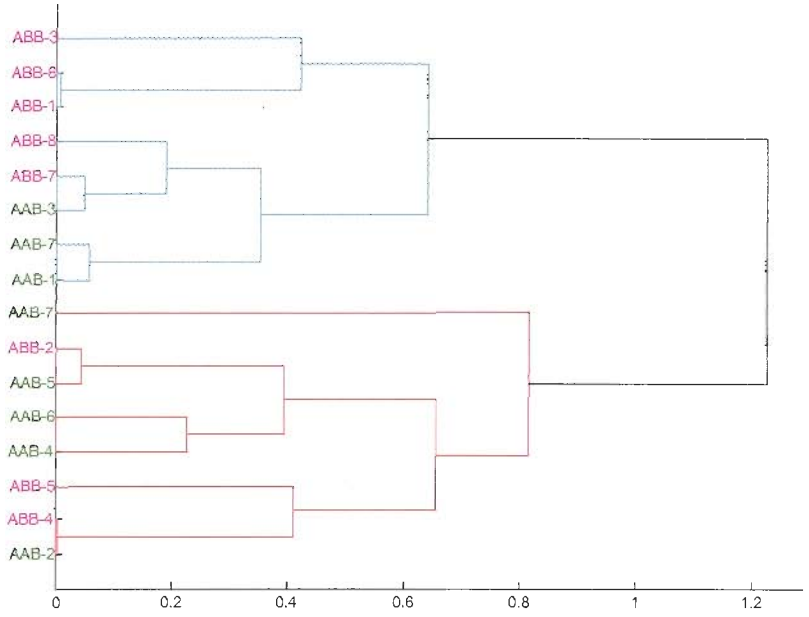

Figure 72 **Results of cluster analysis performed on 16 hidden activation vectors formed at the end of 8 AAB and 8 ABB test sentences, for network #8 (the horizontal values indicate the Euclidian distances between vectors). Cluster 1 (blue) contains 3 AAB and 5 ABB sentences (38% AAB), whereas cluster 2 (red) contains 5 AAB and 3 ABB sentences (63% AAB).**

With regard to simulating Gomez & Gerken's experiment, the cluster analysis performed on the hidden activations formed at the end of each test sentence generated with a finite-state grammar confirms that the model is unable to robustly differentiate between grammatical and ungrammatical patterns. Figure 73 displays the results of the cluster analysis performed on 20 hidden activation vectors formed at the end of 10 grammatical and 10 ungrammatical test sentences, on a network trained to simulate Gomez & Gerken's fourth experiment on infants. Figure 73 also shows, in two different colours, how the network generates two clusters, and how it assigns each hidden activation vector to the two clusters.

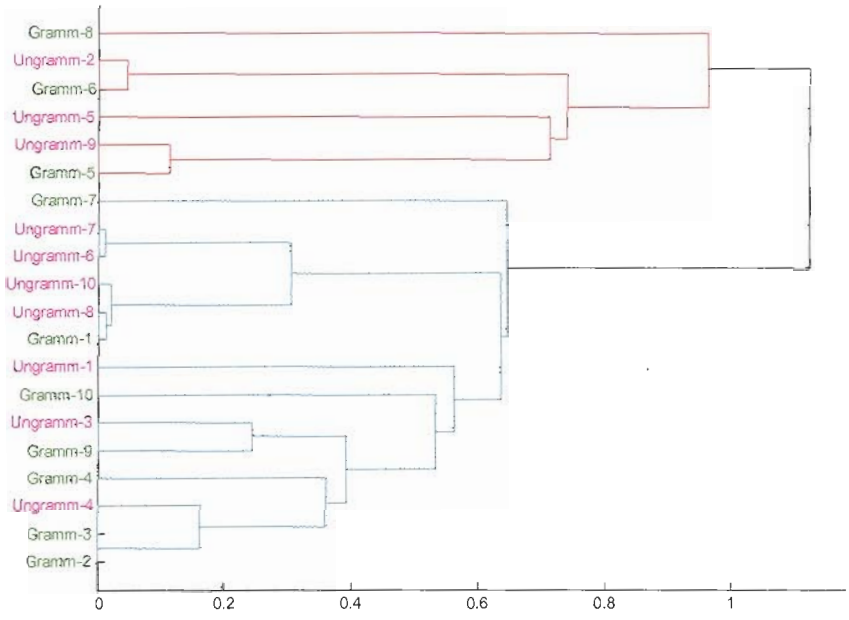

Figure 73 Results of cluster analysis on 20 hidden activation vectors formed at the end of 10 grammatical (Gramm-1 to Gramm-10) and 10 ungrammatical (Ungramm-1 to Ungramm-10) test sentences (the values on the horizontal line represent the Euclidian distances between vectors). Cluster 1 (blue) contains 7 grammatical and 7 ungrammatical sentences (50% grammatical), whereas cluster 2 (red) contains 3 grammatical and 3 ungrammatical sentences (50% grammatical).

Figure 73 illustrates the high level of overlap between internal representations formed by grammatical and ungrammatical test sentences. For example, in cluster 1 (blue), although the internal representations of two ungrammatical test sentences (Ungramm-6 and Ungramm-7) group, they then cluster with the subgroup formed by the grammatical sentence Gramm-1, and ungrammatical sentences Ungramm-8, and Ungramm-10. According to the cluster analysis displayed in Figure 73, there is no subgroup of internal representations that contains more than 2 patterns of the same and only type.

7 NEGISHI'S MODEL

Negishi (1999) uses a variant of a SRN to replicate Marcus *et al.*'s (1999) experiments, and claims that the kind of statistical methods he employs is able to capture Marcus *et al.*'s stimuli.

7.1 Network Architecture and Input Representation

The model used by Negishi (1999) is a simple recurrent network without hidden units, i.e., all the output activations feed back directly to the input layer (see Figure 74).

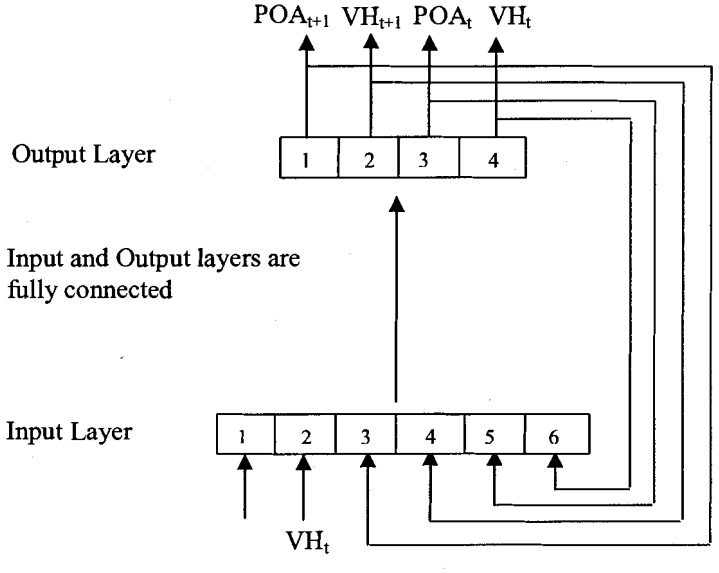

Figure 74 Negishi's model (1999). It is a simple recurrent network without a hidden layer (the activations of output units feed back to the input layer). POA_t and VH_t represent the place of articulation of the consonant, and the height of the vowel contained in the current syllable. POA_{t+1} and VH_{t+1} represent the place of articulation and vowel height of the next syllable.

The network has six input nodes, four output nodes, and no hidden nodes. Two of the input units receive input features, and the other four input units receive feedback from the output layer. The two input features are the vowel height (VH) and the place of articulation of consonants (POA), and represent one syllable at a time (as mentioned earlier, all syllables used by Marcus *et al.*, 1999, consist of a single consonant followed by a single vowel). For example, syllable *ba* is presented to the network using the place of articulation of consonant *b*, and the height of vowel *a*.

Depending on their height, Negishi classifies all English vowels into three categories: low (e.g., letter *a*), middle (e.g., letters *e, o*) and high (e.g., letter *i*), and associates a value between 0 and 1 to each of these three classes: low=0, middle=0.67, and high=1. In speech, consonants can have different places of articulation, which linguists define as the relationship between the active and passive articulators as they shape or impede the airstream. In other words, consonants can be distinguished according to the location of their production in the humans' vocal tract[35]. Negishi uses a value between 0 and 1 to represent all the consonants, depending on their place of articulation in the vocal tract (Giegerich, 1992). For example, *k*=0.125, *g*=0.125, *d*=0.725, *t*=0.725, *b*=1, *p*=1, etc.

Examples of words (syllables) and their representation:

ga: POA=0.125, VH=0

li: POA=0.725, VH=1

[35] For example, consonants can be articulated by the use of the lips (like *m, p*), or by raising the tip of the tongue (like *d, t*), etc. It is unclear, however, how the place of articulation can accurately model the way the infants in Marcus *et al.*'s study attend to the input stimuli, since infants could pay attention to many other (arguably, more relevant) phonetic features of consonants (e.g., palatal, lateral, sonorant, etc).

de: POA=0.725 VH=0.67

ji: POA=0.25, VH=1

7.2 Training, Testing, and Results

The input sentences are presented one word (syllable) at a time, and the network is trained to predict the next word in the current sentence (with two of the output units, marked POA_{t+1} and VH_{t+1} in Figure 74), and to mirror the current input features at the output units (using the other two output units, marked POA_t and VH_t in Figure 74). The input and output layers are fully connected (see Figure 74).

Negishi performed Marcus *et al.*'s (1999) experiments on this modified simple recurrent network and measured the difference between the network's predictions and the expected outputs. He reports that the network's error is significantly lower for consistent test patterns (generated with the familiar grammar) than for inconsistent test sentences (generated with the unfamiliar grammar). Therefore, he claims, "the simulation results show that a type of statistical learning method captured regularity in Marcus *et al.*'s stimuli" (Negishi, 1999).

7.3 Marcus' Evaluation

In his analysis of this model, Marcus (2001) acknowledges that Negishi may provide a valid account of his results on infants. However, he believes that the reason for Negishi's result is the fact that the model relies on using nodes as variables, and the network operates over those two variables. In other words, according to Marcus, this model *implements* a classical mechanism (operation over variables), rather than being a genuine eliminative network.

I find a few problems with this Marcus' analysis. First of all, as mentioned earlier, I think that the way Marcus makes use of the term "variable" is tendentious. The very fact that one can locate input nodes that may represent variables does not indicate how the network internally processes those input representations. As Marcus argues, it is how the network internally manipulates those "variables" which dictates the nature of a network with regard to being eliminative or implementational. Secondly, Marcus does not clearly identify the type of operations over variables that Negishi's model presumably performs[36] and the realm where these operations may have occurred.

According to Marcus, in Negishi's network, "each word is encoded by means of two variables" (Marcus, 2001), and the network connections implement operations that "apply to all instances of a class" (Marcus, 2001). If we assume that the "variables" that Marcus refers to are place of articulation of consonants (POA), and vowel height (VH), then the "class" is *current-word* in each sentence. Based on Marcus' definition, a symbol (which, in this case, represents a pair of variables) is a context-independent representation of the class *current-word*. However, at a closer look, the network cannot apply the *same* operation over all instances of the class *current-word*, because that operation depends not only on the current word, but also on the network's representation of the previous word. In other words, the operation is *dependent* on the context of *current-word*. Based on the structure of Negishi's network (Figure 74), the activation of the output units is determined by three pieces of input data: the current word's input representation, the network's representation of the previous word, and, finally, the network's representation

[36] Since all input values for both POA and VH vary between 0 and 1 (during both training and testing), the main task that the network performs is interpolation between the known values. According to the first Marcus' definition of training space, Negishi's model does not generalize outside the training space. However, according to the second definition, this model does perform generalization outside the training space (which is also true in the Shultz & Bale's case). Apparently, Marcus failed to acknowledge this fact.

of the current word. These three pieces of information, which are part of the input representation, are context-dependent, and this fact casts doubt upon Marcus' claim that the network applies the same operation to all instances of the *current-word* category.

Another possibility is that the "variables" that Marcus refers to are *current-word* and *previous-word*, and the "class" represents the combination of current and previous words (the *current-word & previous-word* category). Considering that the network output at any time depends only on the current and previous words, the network seems to apply the same operation to all instances of the *current-word & previous-word* category. However, this fact alone does not render Negishi's model implementational. The network does learn a certain function that applies to both the current and previous words, but, as shown in the next two sections (Personal Investigation and Knowledge Representation Analysis), this function depends on very specific characteristics of the first word in each *training* sentence (e.g., the difference between the numerical representation of the vowel and the numerical representation of the consonant contained in the first word). If these characteristics are not found in *test* sentences, the function that the network applies fails to generate results that are consistent with those obtained during training. Therefore, I argue that the network does not learn an *abstract* relationship between the *current-word* and *previous-word* variables, i.e., it does not apply the same operation to all *instances* of the *current-word & previous-word* category.

7.4 Personal Investigation

In my own experiments on Negishi's model I was able to reproduce his original results. I trained a large number of networks (over 30), and the model seamed to robustly replicate Marcus *et al.*'s reported results (I used the same network simulator as Negishi).

Unfortunately, problems arose when I started altering the input set. All test sentences contain the vowels *a* and *o* (like in *ba po ba*, or *ko ga ga*, etc). By switching those two vowels between each other in all tests sentences where they occur (for example, *ba po ba* became *bo pa bo*, and *ko ga ga* became *ka go go*, etc), and repeating all experiments in (otherwise) the same conditions, I discovered that the networks consistently fail to make the same discrimination as before (lower errors for familiar test sentences than for unfamiliar ones). Granted, I do not know what infants would have done in these conditions, but to an adult the changes seem minimal and manageable. In any case, this casts doubt upon the robustness of Negishi's model, and shows that it lacks even the simplest generalization capabilities.

I also tested the model's ability to deal with more complex grammars (such as the finite-state grammars shown in Figures 5 and 6). I used the same input sentences as Gomez & Gerken (1999): 10 training sentences formed with the grammar shown in Figure 5 (the training sentence have lengths between 3 to 6 words), 10 novel test sentences formed with the familiar grammar, and another 10 test sentences formed with the unfamiliar grammar (Figure 6). I trained 32 different networks in these conditions, and none of the network is able to correctly discriminate between familiar and unfamiliar test sentences.

Similar to Shultz (1999), and Shultz & Bale (2001), Negishi's model seams to be able to replicate the results reported by Marcus *et al.* (1999), but when tested with slightly changed stimuli, or with more complex grammars, it shows a lack of robustness and generalization capabilities. This suggests that the training algorithm does not thoroughly make the networks learn the underlining syntactic categories of the input

patterns, which is essential in order to generalize beyond the limited domain of the input data. This theory is confirmed by my knowledge representation analysis.

7.5 Knowledge Representation Analysis

One of the particularities of this SRN is that it does not have any hidden layers: the outputs feed back directly to the input layer. The network is presented with one syllable at a time (a pair of one consonant and one vowel), and is trained to predict the next syllable in the sentence (in two of the four output units), and also to mirror the current input syllable at the output layer (in the other two of the output units) (see Figure 74). However, when measuring the network's performance, Negishi only uses the first two output units (i.e., only the ability to predict the next word is measured). More specifically, the network's performance is given by its ability to predict the third word of the sentence, i.e., only the first two words of the 3-word sentences are ever presented to the network (even during training).

Since only the outputs of the first two output units (those which predict the *next* syllable) are used to measure the network error, I focused my analysis on how those two output values are computed. Additionally, in the second and third of Negishi's experiments, the consonants that appear on the "A" position have the same numerical representation as the consonants that appear on the "B" position (e.g., "b" and "p", and "k" and "g", respectively). This means that the network output associated with the next consonant (POA_{t+1}) does not influence the way the network differentiates between ABA and ABB structures (because that output will be the same for both ABA and ABB patterns). Thus, in the second and third experiments, the only factor that affects the

network's ability to differentiate between the two grammatical structures is how well it can predict the next *vowel* in the sentence (which is the second output unit: VH_{t+1}).

7.5.1 Connection weight analysis

A comparative view of the connection weights developed by the network following ABA training in experiment 2 is shown in Figure 75. Black boxes denote negative weights, whereas white boxes denote positive weights. The relative size of each box indicates the value of the weight (the bigger the size, the stronger the weight). The numbers above each box indicate the input unit (from 1 to 7, where 7 is the bias unit).

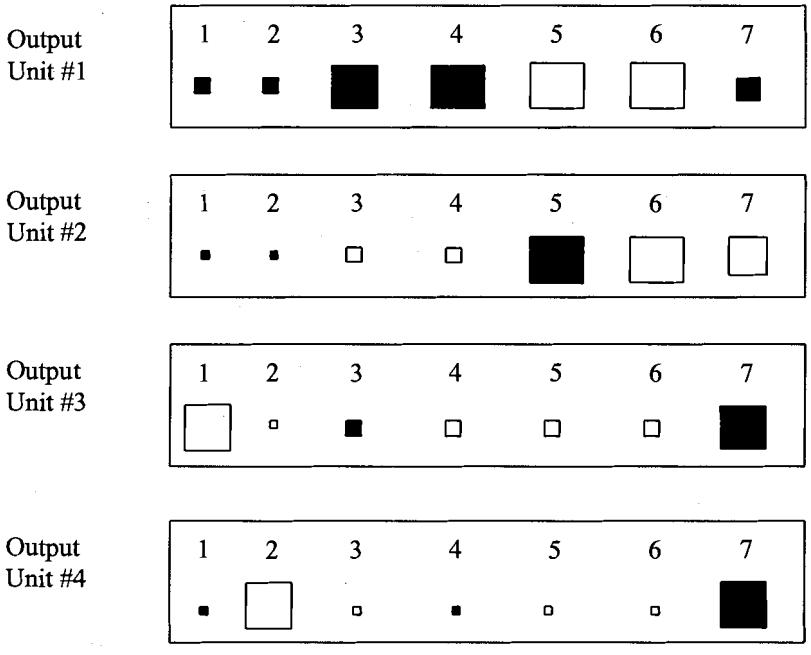

Figure 75 The distribution of the connection weights in Negishi's model following ABA training in experiment 2. The numbers above each box represent the input unit (from 1 to 7, where 7 is the bias unit). Black boxes indicate negative weights, whereas white boxes indicate positive weights. The relative size of each box indicates the strength of the weight.

Observing the connection weights developed by the second output unit following ABA training (we already established that the first output unit is not relevant with regard to the network error in experiments 2 and 3), I noticed that the most significant weights are towards the input units #5 (highly negative value) and #6 (highly positive value). All the other connection weights developed by this output unit are either at least three orders of magnitude smaller, or are constant (e.g., towards the bias unit[37]). This suggests that the output of this unit depends on the difference between the network prediction for the vowel (input unit #6) and the network prediction for the consonant (input unit #5) of the previous syllable. Since only the first two words of any sentence are presented to the network, and since the performance is measured after the second word, the network's ability to differentiate between ABA and ABB patterns depends entirely on the difference between the vowel and consonant representations of the first "A" word. Fortuitously, the "A" values that are part of the original test vectors used by Negishi (which are based on Marcus *et al.*'s stimuli) happen to generate "good" results. However, we can always find a consonant and a vowel whose difference in input representations is quite dissimilar from those presented during training. For example, by just switching the words between the first and second positions ("po ba po" instead of "ba po ba"), the network error is smaller for ungrammatical test sentences than for grammatical ones (0.42 vs. 0.53). Granted, I do not know what infants would do in this situation, but it is highly unlikely they would fail to differentiate the new, slightly different structures.

[37] The connection weights developed between the bias unit and the first and second output units are equal to the average numerical representation of all consonants, and the average numerical representation of all vowels, respectively.

7.5.2 Principal component analysis

PCA of contributions (products between hidden activations and output weights) at the end of each sentence (i.e., after the second word of each sentence is presented) unfolds a similar situation. It reveals 3 principal components, which account for about 99.4% of the variation. According to Figure 76, the first principal component (50% of the variation) separates the network contributions based on the vowels that appear in the input patterns (two clusters, one for "e", and one for "i"). The next principal component (40% of the variation) separates the network contributions generated by the middle word based on the first word of each sentence (in particular, as mentioned above, this dependency is based on the difference between the vowel and consonant values of the first word). In Figure 76 the first word is specified in parenthesis. Letters "d" and "l" have the same input representations[38], so their contributions are identical. Figure 77 displays the third principal component (9% of the variation), which separates the network contributions into three groups, based on the consonant values of the middle word: one for letter "w", one for letter "j" (very close to the first cluster, since their input representations are close), and another one for letters "d" and "l" (they have the same input representation).

[38] According to Negihi's input representation, consonants "d" and "l" are represented by the same value: 5/8.

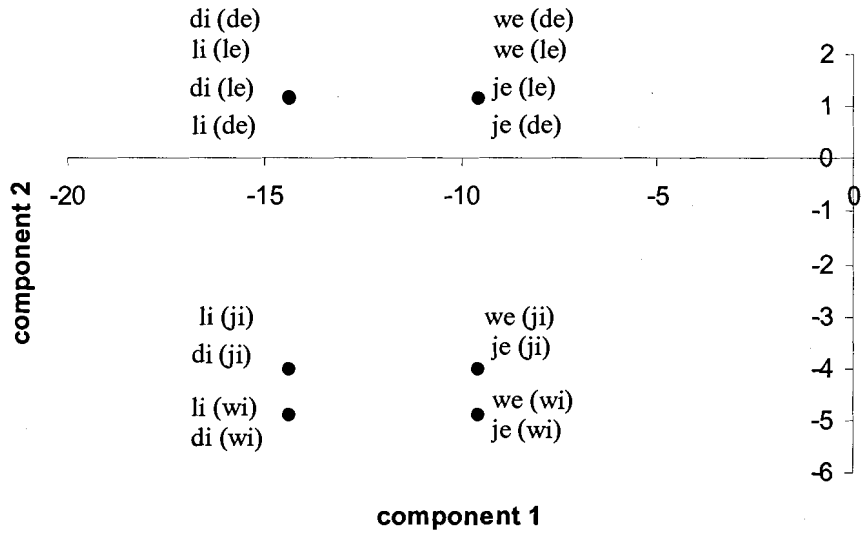

Figure 76 **Projections of network contributions onto the first two principal components (with regard to the middle word and the first word, in parenthesis, of each sentence). Along the first principal component, the network contributions are separated based on the vowels that appear in the middle word ("e" and "i"). Along the second component, the contributions are separated based on the words that appear on the first position on each sentence.**

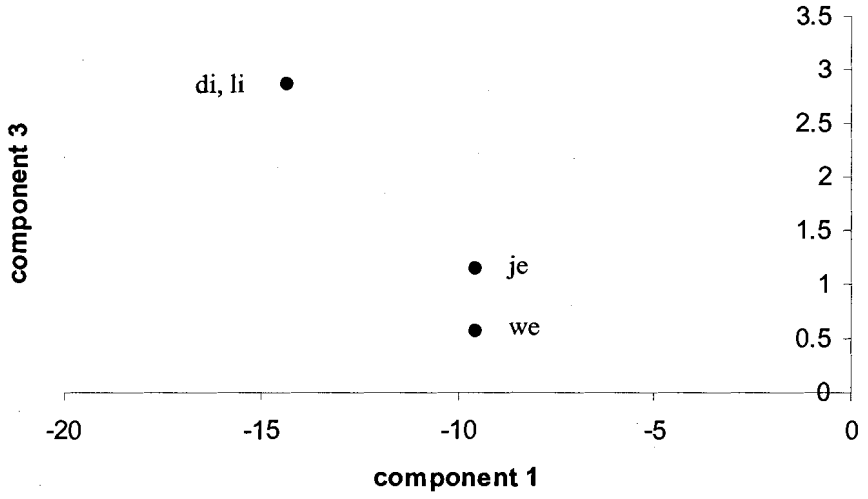

Figure 77 **Projections of network contributions onto the first and third principal components (with regard to the middle word of each sentence). Along the third component, contributions are separated based on the middle word of each sentence.**

Below, Figures 78 and 79 display the projections of *network contributions* onto the first three principal components, generated by *each* input pattern (the first and second words of each sentence). The same pattern of network contributions emerges: the first principal component separates the contributions onto two clusters based on the vowels contained in the input vectors. The second principal component separates the contributions based on the previous word (for the first word of each sentence, the previous word is null, and this is why their contributions are null with regard to the second principal component). Finally, the third principal component separates the contributions based on the consonants contained in the input words. Significantly, according to PCA of contributions (which consist of hidden activations and output weights), most of the network's resources (99.4%) are used to encode very specific information about the particularities of the training patterns (such as the types of vowels, consonants, and the difference between the representations of vowels and consonants). The network does not allocate its resources to abstracting the grammatical structure of the input stimuli, which is essential in order to robustly generalize to novel input and deal with more complex grammars.

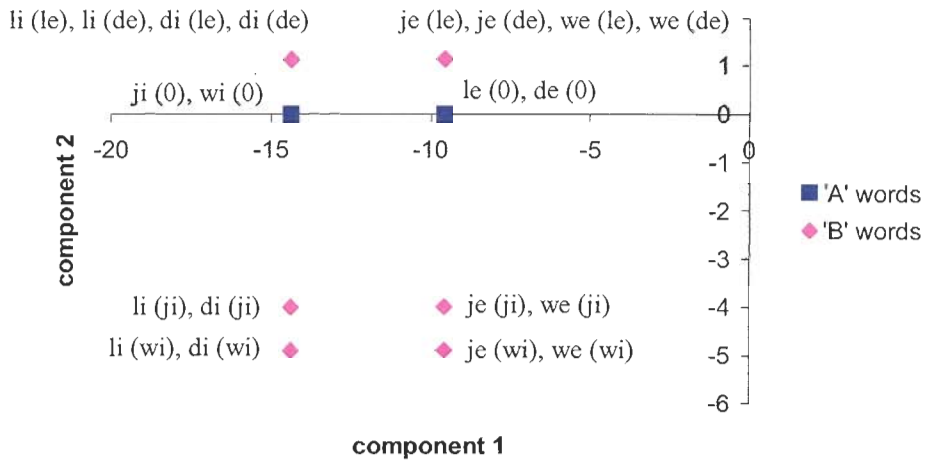

Figure 78 **Projections of network contributions onto the first two principal components (with regard to both the first and second words of each sentence; the previous words are specified in parenthesis). There is a clear demarcation between words based on their vowels (first principal component), and based on the previous words in each sentence (the second principal component).**

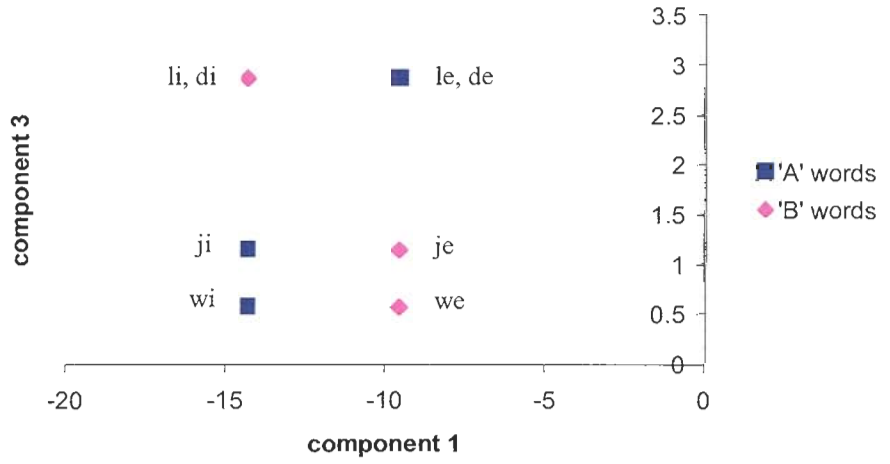

Figure 79 **Projections of network contributions onto the first and third principal components (with regard to both the first and second words of each sentence). Along the third principal component, the network contributions separate based on the consonants that appear in the input words.**

According to the analysis performed up to this point, it is clear that the network will have problems handling more complex (Gomez & Gerken) grammars. In this case, PCA of contributions at the end of each sentence reveals two principal components (which account for about 97% of the variation). According to Figure 80, the first principal component (74% of the variation) separates the network contributions based on the vowels in the words that appear at the end of each sentence. The second principal component (23% of the variation) separates the network contributions based on the consonants in the words that occur at the end of each sentence. Similar to the case of simpler grammars, the network allocates most (if not all) of its resources to identifying very specific information about the particularities of the training patterns. It is not able to do any abstraction of the syntactical structure of those patterns, and this may explain why it fails at this task. Anyway, the fact that the network output at any time can only depend on the current and previous syllables suggests that it may have problems dealing with longer sentences. The network can only predict the next word based on two previous words. The network's structure makes it impossible to deal with more complex grammars where the position of a certain word within a sentence depends on more than two previous words[39].

[39] However, the network structure can be altered to deal with *certain* complex grammars.

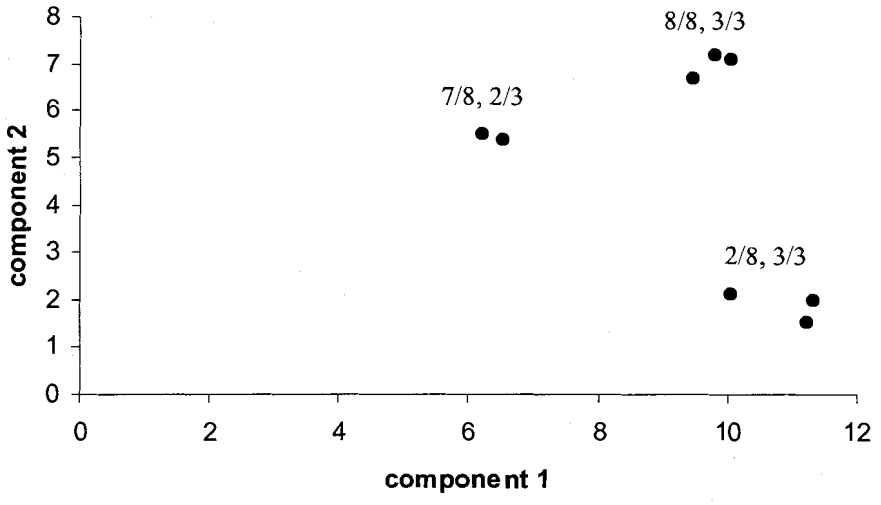

Figure 80 **Projections of network contributions onto the first two principal components (with regard to the numerical representations of consonants and vowels occurring in the words at the end of each sentence). The first number indicates the numerical representation of consonants (i.e., the place of articulation), whereas the second number indicates the numerical representation of vowels (i.e., vowel height). Along the first principal component, network contributions separate based on the numerical representation of vowels, whereas, along the second principal component, contributions separate based on the numerical representation of consonants.**

In order to see how the network groups the Gomez & Gerken test stimuli, a cluster analysis has been performed on the 20 hidden activations formed at the end of the 10 grammatical and 10 ungrammatical test sentences. Figure 81 shows the results of this analysis, along with information regarding how the network creates two clusters and how it assigns the hidden activations to one of the two clusters.

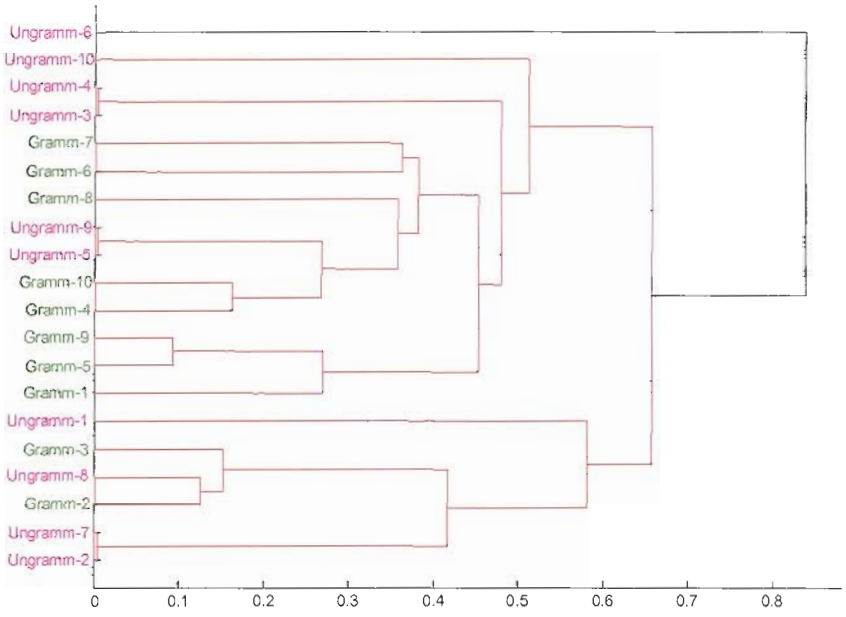

Figure 81 **Results of cluster analysis on 20 hidden activation vectors formed at the end of 10 grammatical (Gramm-1 to Gramm-10) and 10 ungrammatical (Ungramm-1 to Ungramm-10) test sentences (the values on the horizontal line represent the Euclidian distances between vectors). Cluster 1 contains only 1 ungrammatical sentence (100% ungrammatical), whereas cluster 2 (red) contains 10 grammatical and 9 ungrammatical sentences (47% ungrammatical).**

According to Figure 81, there is a high degree of overlap between hidden activations formed by grammatical test sentences and hidden activations formed by ungrammatical test sentences. For instance, in cluster 2 (red), although activations formed by grammatical sentence Gramm-6 groups with Gramm-7, and those formed by Gramm-4 groups with Gramm-10, these two subgroups then cluster with activations formed by ungrammatical sentences Ungramm-5 and Ungramm-9. None of the subgroups in cluster 2 contains more than 3 internal representations formed with the same (and only) grammar.

8 IMPLEMENTATIONAL CONNECTIONIST MODELS

This section lists three attempts to replicate Marcus *et al.*'s (1999) study using *implementational* connectionist models, i.e., the models implement some form of classical mechanisms (e.g., variable binding, operations over variables) within a neural network.

8.1 Hand-wired Network

Gasser & Colunga (1999) attempt to replicate Marcus *et al.*'s (1999) experiments with a hand-wired network called Playpen (Gasser & Colunga, 1997). This model employs so-called "relation units" (Gasser & Colunga, 1999), which are handcrafted clusters of basic units. For the purpose of simulating Marcus *et al.*'s experiments, Gasser & Colunga define two types of relation units: sameness units (clusters of input units that are associated with similar words), and difference units (clusters of input units that are associated with dissimilar words). These relation units are part of the so-called "correlations" layer (Gasser & Colunga, 1999).

The input layer is divided into three sections, each section with its own set of input units (a section corresponds to a position of a word within a sentence). All three input sections have the same number of units, each unit corresponding to a word (localist representation). In each of the three input sections there is an additional, generic input unit called CV (consonant-vowel).

In the correlations layer, Gasser & Colunga create sameness and difference units between pairs of input units from all three input sections (the two input units that are linked to a relation unit are in *different* input sections). For example, if two input units encode the same word in two different input sections, a sameness unit is created and hand wired to those two units. If two input units from different sections encode different words, a difference unit is created and hand wired to those two units. Sameness and difference units are also created between each pair of CV units. The units on the correlations layer are fully connected to each other.

Citing Elman's work (Elman, 1999; Seidenberg & Elman, 1999), Gasser & Colunga argue that this kind of neural structure simulates the knowledge of "syllable similarity" (Gasser & Colunga, 1999) that the infants might have before participating in Marcus *et al.*'s (1999) study.

The sentences are presented to the network one at a time, by activating the input units that correspond to the three words of the current sentence in each of the three input sections. All CV units are also activated, as well as all the corresponding sameness and difference units on the correlations layer. For example, when presenting the sentence "le le di", the unit associated with the word "le" in the first set of input units (the one corresponding to the first position), the unit associated with the word "le" in the second set (the one corresponding to the second position), the unit associated with the word "di" in the last set (the one corresponding to the last position), and all three CV units are activated. Also, the sameness unit hand wired to the input units corresponding to the word "le" in the first and second positions, as well as two difference units hand wired to the input units that correspond to words "le" and "di" in the first and third positions, and

to those that correspond to words "le" and "di" in the second and third positions, respectively, are all activated. In addition, one sameness unit hand wired to the first and second CV units, as well as two difference units hand wired to the first and third CV units, and to the second and third CV units, respectively, are all activated.

Following the activation of all these units that correspond to the current training sentence, Gasser & Colunga apply a learning procedure called Contrastive Hebbian Learning (CHL)[40] (Movellan, 1990) that causes the strengthening of all connections between active units and the weakening of all connections between inactive units, both within and towards the correlations layer. Following training, Gasser & Colunga test their network with novel sentences created with both the training grammar, and another, unfamiliar grammar. They show that the test patterns that are formed with the training grammar generate "more activation" (Gasser & Colunga, 1999) on the correlations layer (i.e., the sum of activation values of all relation units), than the test patterns that are inconsistent with the training stimuli. Based on this result, Gasser & Colunga claim that they successfully replicate Marcus *et al.*'s (1999) study.

This network is, however, a clear example of an implementational model that performs variable binding. In this case, the variables are the positions of words in the input sentences, and the relation units "bind" the instances of those variables based on their similarity. Most (if not all) of the knowledge is provided by the external supervisor, who hand wires both the relations between words, and the relations between the position

[40] Contrastive Hebbian learning is a Hebbian-type algorithm (Hebb, 1949) where the connection weights between two units are strengthened when both units are active. The connection weights are weakened when neither of the input units is active. Contrastive Hebbian learning is a supervised algorithm, and has two phases (states). In the first phase (the so-called "free state"), the input units are activated, and the Hebbian learning rule is applied to each pair of connections. In the second phase (the so-called "clamped state"), the output units (the relation units in the case of Gasser & Colunga's model) are activated and the connection weights are updated based on the Hebbian learning rule.

of those words within sentences. In effect, the external supervisor handcrafts the syntactical structure of the training patterns.

Although this model may be able to replicate the results reported by Marcus *et al.* (1999), it is questionable whether it can handle more complex grammars. First of all, in the current design, the Playpen model can only handle input patterns of the same length. But, even if the design is changed to support variable length patterns, the sameness and dissimilar rational units may not be enough to capture the higher variability in the finite-state grammars. For example, in a certain grammatical sentence, a word *w1* on position *p1* (denoted *w1/p1*) may be the same as word *w2* on position *p2* (denoted *w2/p2*). This can be captured by one sameness unit between *w1/p1* and *w2/p2*, and another sameness unit between the CV units corresponding to *p1* and *p2*. But, in a different grammatical sentence (using the same grammar), the words on positions *p1* and *p2* may not be similar anymore, which means that a dissimilar unit needs to be created between the CV units corresponding to *p1* and *p2*. As a result, those CV units will be part of both sameness and dissimilarity relation units, and this conflicts with the very design of the Playpen architecture.

8.2 Shastri & Chang's Model

Shastri & Chang (Shastri, 1999; Shastri & Chang, 1999) constructed a recurrent neural network that explicitly implements algebraic rules within neural substrate. The characteristics of their model are (see Figure 82):

- One set of nodes encodes the position of a word within a sentence (e.g., *first word*, *second word*, *third word*). These nodes are called *positional role* nodes (denoted

P1, P2, and P3, indicating the first, second, and third position within a sentence), and act as both input and output nodes.

- Another set of nodes encodes each word based on a distributed representation having 12 phonetic features (e.g., voiced, nasal, etc). These are called feature nodes, and act as both input and output nodes.

- The network contains two fully connected hidden layers. The positional role nodes and feature nodes are both connected to the first hidden layer, whereas the activations of the second set of hidden units feed back to the positional role nodes and feature nodes.

- The network binds a positional role node with an input word by synchronously activating the positional role node that specifies the word's position within a sentence, and all the feature nodes that encode that word.

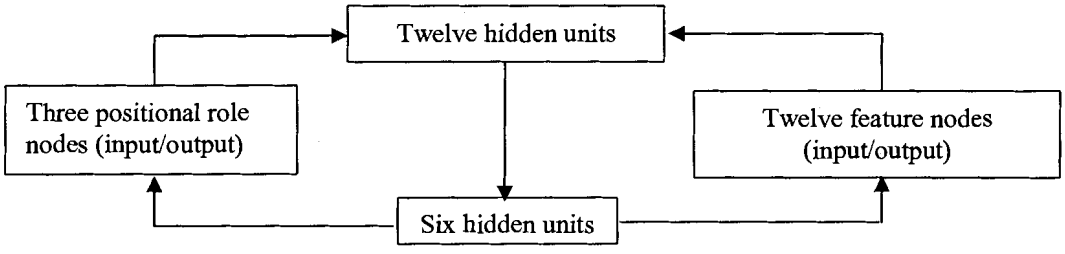

Figure 82 Shastri & Chang's (1999) time synchronous network. The feature nodes encode the current word, whereas the positional role nodes encode the position of the current word within the sentence. The layers are fully connected to each other.

The training sentences are presented to the network one word at a time, by activating the feature nodes that correspond to the current word, as well as the positional role node that corresponds to the current position of that word within the sentence. For

each word, once the input information propagates through the network, in a typical backpropagation fashion, the difference between the actual pattern of activations of the positional role nodes and the desired activations of those nodes is used to update the connection weights of the network. The connection weights are modified in order to minimize this difference for each training sentence. Based on this procedure, Shastri & Chang train their network to "detect *coincident* activity" (Shastri & Chang, 1999) of the input/output nodes. For example, the network learns the ABA grammar by learning that the positional role nodes P1 and P3 *coincidentally* fire. The connection weights linking the second hidden layer to the first and third positional nodes are very similar, and this means that they receive similar amounts of input activation. Therefore, whenever P1 fires (as a result of presenting a word in the first position of a sentence), P3 automatically fires as well.

During testing, the positional role nodes and feature nodes are activated for each word in the same fashion as during training. Following the propagation of the input information through the network, the activations of positional role nodes for each input word (i.e., the position that the network predicts for the current word) are compared to the target activations of those nodes (i.e., the real position of the current word within the test sentence). Shastri & Chang found that the mean squared difference between the network's predictions of word positions and the actual positions is lower for test sentences that have the same form as the training patterns than for test sentences that have a different (unfamiliar) form.

However, the network does rely on an external supervisor that provides very important feedback regarding the position of each word within a sentence. This external,

explicit information is essential in order for the network to succeed at this task. In effect, the external supervisor teaches the network that one positional role node acts the same as another positional role (for example, during ABA training, the network is explicitly taught that the first positional role node fires at the same time as the third positional role node), i.e., the external supervisor explicitly teaches the network the repetitional structure of the input patterns. In a way, these positional role nodes can be considered "temporal variables" (Marcus, 2001) that are instantiated with the current position of a word within a sentence (*first-word*, *second-word*, *third-word*). Because of this, it does appear that the network implements some sort of classical mechanism, by binding those temporary variables to the same instantiations (for example, the instances of *first-word* are the same as the instances of *third-word*).

Although Shastri & Chang are able to successfully replicate all the experiments performed by Marcus *et al.* (1999) on infants, it is unlikely that this model can handle more complex grammars. When using grammars similar to those by Gomez & Gerken (1999), the position of words within sentences varies. For example, in a certain grammatical sentence, the words in two different positions may be similar, but in another grammatical sentence, the words in those two positions may be different. It is unclear how Shastri & Chang's model can learn a more complex grammar just by learning the coincidental activities of the positional role nodes.

8.3 Dominey & Ramus' Model

In order to replicate Marcus *et al.*'s (1999) experiments, Dominey & Ramus (Dominey & Ramus, 2000) use a slightly changed version of a temporal recurrent network (TRN) called abstract recurrent network (ARN).

A TRN is similar to a simple recurrent network, but has a few differences. Firstly, the connection weights between the context and hidden layers, as well as those between the input and hidden layers are fixed (i.e., they do not change during training). Only the connection weights between the hidden and output layers are allowed to change. The hidden layer encodes a so-called "internal state", which represents the position of the current element in a temporal sequence, as well as the duration of the sequence and the delay between elements. Learning is performed using a simple associative learning mechanism, which modifies the connection weights between the hidden and output layers in order to associate the internal state (the contents of the hidden layer) with the current output. Each connection weight between the active output unit[41] and the hidden units that encode the current state is updated with the product among a fixed learning rate, the activation of the output unit, and the activation of the hidden unit (similar to the Hebbian learning method). All nodes in the network are "leaky integrators" (Dominey & Ramus, 2000), i.e., they do not instantaneously become active. Instead, they are characterized by so-called charge and discharge times, which are the number of time steps needed by a node to activate and deactivate, respectively. At each time step, one element of the temporal sequence is presented to the network, and the network is trained to output the same item that is presented at input. The performance is dictated by the output nodes' response time, which is, the number of time steps needed to activate the output node that corresponds to the current input element. According to Dominey & Ramus, the logic behind this training procedure is that, during repeated presentations of the same sequential structure, there results a certain pattern of activity in the hidden layer that is

[41] The network uses localist representations on both input and output layers, i.e., only one input and one output unit is active at each time step.

common to that sequential structure. As a result, the response times of the output nodes associated to temporal sequences that have the same structure are reduced. Dominey & Ramus claim that the TRN model is developed in order to explain the electrophysiological recordings of neurons in the prefrontal cortex of monkeys that perform temporal learning tasks (Barone & Joseph, 1989).

When simulating Marcus *et al.*'s (1999) study, Dominey & Ramus discovered that a TRN is unable to reproduce the results reported by Marcus *et al.*, because the model cannot generalize its knowledge to sequential structures formed with novel elements. In order to address this problem, Dominey & Ramus propose a slightly changed version of TRN called abstract recurrent network (ARN). The only difference between TRN and ARN is the fact that the latter does not use the current item in the temporal sequence as input. Instead, an *abstract* representation of the *entire* sequential structure is used as the network's input. This abstract representation is encoded on a special layer of units, called recognition layer, which is inserted between the input and hidden layers. This layer uses a recognition function to compare the current input element to up to 5 previous input elements (which are stored separately into a so-called working, or short-term memory, STM). The recognition layer contains 6 units. The first unit is activated when the current input is the same as the previous input. The second recognition unit is activated when the current input item is the same as the input element presented before the previous input, etc. If none of the previous 5 input elements matches the current input item, the sixth (and last) recognition unit is activated. Once the entire sequential structure has been encoded within the recognition layer, the training is conducted in the same way as described above for TRN. Since all training sentences in

the Marcus *et al.* study have the same sequential structure (e.g., ABA, or ABB), the Dominey & Ramus network will always be presented with the same input pattern, namely, an abstract representation of the sequential structure of the training sentences. The training procedure will therefore minimize the response time of the one and only output node that corresponds to the unique input pattern. Dominey & Ramus found that, when presented with novel sequences of words, the response time of the node that corresponds to the new sequential structure is greater than the response time of the node that corresponds to the training structure. Using this procedure, Dominey & Ramus successfully replicate all three experiments of Marcus *et al.* (1999).

However, as Marcus (2001) argues, this is a clear example where a classical mechanism is employed. In a typical classical algorithmic fashion, the network stores up to five previous input items in a short-term memory and, at each time step, compares the current input with the previous five. The network does not *learn* this procedure. Instead, the entirely classical mechanism is handcrafted into the network[42], and represents the main reason for its success.

Dominey & Ramus' (2000) model may be suitable for more complex grammars (similar to those used by Gomez & Gerken, 1999). However, the size of the short-term memory may need to be adjusted accordingly. In the current configuration, Dominey & Ramus' model can handle grammars where the position of an item within a grammatical structure can be uniquely determined by up to 5 previous items. If the high-order

[42] It is noteworthy that, similar to Elman's simple recurrent network (Elman, 1999; Seidenberg & Elman, 1999), an external supervisor makes the decisions on the sameness of words. It is unlikely that the infants in Marcus *et al.*'s study have any knowledge regarding word similarities before participating in Marcus *et al.*'s study.

dependencies in a grammar exceed 5, the size of the short-term memory (as well as the length of the recognition layer) will need to be increased.

9 DISCUSSION AND CONCLUSIONS

9.1 Classicism vs. Eliminative Connectionism

The kind of mental representations functioning in the human mind has been the subject of debate between classicists and eliminative connectionists since the late 1980s. On one hand, classicists believe that the mental representations in the human mind are characterized by complex syntax and semantics, and that the mental processes operating over those representations are dictated by their syntactic structure. Classicists argue that, in order to model cognition, the cognitive architectures need to involve operations on explicitly structured symbolic representations (Fodor & McLaughlin, 1990; Fodor & Pylyshyn, 1988; Newell, 1980). On the other hand, eliminative connectionists believe that the mind is a system composed of simple processing elements (nodes) and connections (that resemble the biological neurons and synapses in the human brain), which can exhibit intelligent behaviour without involving operations on symbols and variables. Eliminative connectionists argue that this paradigm is more plausible than the classical theory with regard to simulating cognitive processes.

Supporting the classical point of view regarding the type of mental representations in the human mind, in a series of studies, Marcus (1998, 1999, 2001) argues that symbols, variables, and operations over these variables represent the underlying features of the human mind, and they should constitute the basis for all scientific theories regarding cognition. He claims that a very important set of eliminative connectionist models (those trained with the backpropagation algorithm) are unable to

explain certain cognitive processes that require generalization outside the space of the training examples.

In a very well known study, Marcus *et al.* (1999) show that 7-month-old infants are able to differentiate between test stimuli formed with two different grammars (such as ABA vs. ABB). Since the test stimuli are novel with regard to the infants' habituation phase, and since, according to Marcus, backpropagation neural networks cannot generalize outside the space of the training examples, Marcus *et al.* (1999) claim that those networks will be unable to perform the same task as infants.

9.1.1 Generalization outside the training space and training independence

Marcus argues that the main reason for the inability of backpropagation neural networks to generalize outside the training space is the existence of the so-called training independence, which presumably occurs in those networks. In essence, according to Marcus, "training independence" means that the training of a node in the network does not depend on the training of any other node that resides on the same layer. Although the way in which backpropagation is conducted may suggest that networks are susceptible to training independence, in reality, there are ways to minimize (and even eliminate) its effects (e.g., using sufficiently rich and well-defined distributed representations). As argued during the analysis of Elman's network (Elman, 1999; Seidenberg & Elman, 1999), even if the training of one node does not *directly* influence the training of other nodes on the same layer, it does have an effect on the training of all the nodes on the onward layers. Because of this, the training of that particular node *indirectly* affects the training of all the other nodes that reside on the same layer. Also, there are cases where

training independence does not occur at all. For instance, it is questionable whether this effect ever occurs in competitive networks (Hadley, Arnold, & Cardei, 1998).

In my opinion, the significance of training independence that Marcus refers to is overstated. As shown in this thesis, for several connectionist models, such as Shultz (1999), Shultz & Bale (2001), and Negishi (1999), training independence is not an important factor. Although they are trained with backpropagation[43], each one of those models is able to mirror the *reported* behaviour of infants in Marcus *et al.*'s study, and generalize to novel input in the same way as demonstrated by Marcus *et al.* in infants.

9.1.2 Symbols, variables, and operations over variables

Another issue on which I disagree with Marcus is his definitions of "symbolic" systems and "variables", which I find to be somewhat misleading and tendentious. Marcus states that variables "allow us to extend generalizations to novel instances of categories" (Marcus, 1998), where a category "consists of all instances bearing a given label" (Marcus, 1998). For example, the words *walked*, *ate*, *sang* are instances of the category *verb*. Categories, instances, and variables are all represented by symbols. Marcus uses various definitions for symbols, but in essence, he calls a "system that has context-independent representations of categories *symbolic*" (Marcus, 2001) (Marcus' emphasis). For example, if a neural network is presented with various words, and each word is represented in the same way regardless of its context (e.g., the word *cat* always triggers the activation of the same input units, regardless of its position within a

[43] The cascade-correlation algorithm employed by Shultz (1999) and Shultz & Bale (2001) uses a variation of backpropagation called quickprop (Fahlman, 1988). Quickprop is very similar to backpropagation, except for a dynamically adjusted learning rate (the learning rate typically decreases during training). Fahlman (1988) demonstrates that quickprop may improve the learning speed of a feed-forward neural network.

sentence), then that network makes use of symbols: "since multilayer perceptrons have context-independent representations of categories, I count them as having symbols" (Marcus, 2001).

Although Marcus acknowledges that this definition is "permissive" (Marcus, 2001), he argues that the issue is not whether a network makes uses of context-independent representations (i.e., symbols), but whether the network "represents" (Marcus, 2001) variables and operations over variables. However, it appears that Marcus does not distinguish between variables that are "represented" on the external layers of a network (i.e., one can physically identify nodes or sets of nodes on the input and output layers of a network that represent "variables", according to Marcus' definition), and variables that reside on the internal (hidden) layers of a network. Therefore, with regard to the debate between classicism and eliminative connectionism, Marcus considers that if one can identify "variables" in a neural network, regardless of where and how those variables may be represented, that neural network will necessarily *implement* operations over those variables.

On the contrary, I believe that the mere presence of an identifiable "variable" (in Marcus' sense) on the external layers of a network should not *necessarily* indicate an implementational character (as in an implementation of a classical mechanism) of that network. Instead, we should analyse whether those variables are implemented within the network's *internal* layers. There are at least two arguments that support this statement.

Firstly, many eliminative connectionists view eliminativism as a paradigm that requires the elimination of explicitly structured symbolic representations from a network's internal layers. When analysing the implementational or eliminative character

of a connectionist model, we need to ascertain the existence of symbols or variables within the network's internal representations, because this is what eliminative connectionists claim to eliminate.

Secondly, according to Marcus' definition of symbolic systems (i.e., a network that is presented with context-independent representations of categories is symbolic), any connectionist model whose input or output layers are constructed in such a way that they accept explicit symbolic representations makes use of symbols and variables. This makes virtually all connectionist models implementational. However, eliminative connectionists should acknowledge that symbolic representations are valid as input representations for eliminative networks, since it is generally believed that humans accept explicitly structured symbolic representation as input. Therefore, we should not consider that a network is implementational just because it is presented with classical representations.

Based on these considerations, I argue that when we analyse the existence of "variables" within neural networks, we should study the presence of those variables within the networks' internal representations. In other words, I argue that a connectionist model is implementational only when we can *explicitly* identify "variables" *within* the internal representations of the network. If we cannot identify such variables, I argue that the neural network is *eliminative*. Consequently, when analysing the connectionist models discussed in this thesis, I have closely investigated the nature of their internal representations with regard to implementing variables and operations over those variables. Based on this investigation, I believe that, in several cases, Marcus is mistaken in his evaluation.

9.2 Connectionist "Counterexamples" to Marcus

Elman (1999) was one of the first to propose a connectionist model specifically intended to replicate Marcus *et al.*'s (1999) results. He initially pre-trained a simple recurrent network to distinguish whether a given syllable is identical to a previous syllable. He then trained the same network to discriminate between sequences of syllables generated by two simple grammars (ABA and ABB). Elman claims that his experiment shows that SRNs successfully match the infants' reported results.

In his analysis of this model, Marcus (2001) argues that the apparent success of the network is caused by the presence of the preliminary training phase, and the way this phase is performed. To Marcus, this preliminary training phase looks like an "external supervisor" (Marcus, 2001) that teaches the network an algebraic rule of the form "for all syllables **x** and **y**, if **x**=**y**, then output 1, otherwise output 0". I agree that this external supervisor may implement a rule of that form, and because the supervisor is part of the system, this could make the whole system implement that rule. Elman has not demonstrated how the neural network could internally implement this rule.

In recent work (Vilcu & Hadley, 2001), we performed the same kind of simulation as Elman (1999), using numerous simple recurrent networks and a wide range of training parameters. Our results show that, even with the pre-training phase in place, Elman's claim is premature, and his networks perform erratically. In particular, we trained 64 different SRNs, using the same procedure as Elman (1999), and showed that only a small percentage of networks successfully discriminated between the two grammars. I emphasize, however, that Elman's simulation differs from Marcus *et al.*'s experiment not only by the addition of a pre-training phase, but also by training the

network on both grammars at the same time. It is arguable whether Elman's task is any more difficult than Marcus *et al.*'s infants, but it does make his results less relevant with regard to replicating Marcus *et al.*'s data. However, even if we ignore this flaw, the fact that this network performs inconsistently cannot be overlooked. In my opinion, this undermines Elman's claim that he provided a counterexample to Marcus *et al.*'s theoretical arguments that no eliminative connectionist model is able to differentiate between novel grammatical structures.

The knowledge representation analysis that I have performed on this model shows that the primary task that Elman's simple recurrent network carries out is the learning of very specific characteristics of the training stimuli, such as the various vowels and consonants that occur in those stimuli, rather than extracting the abstract structure of the input sentences. According to the principal component analysis of the hidden activation vectors formed at the end of each training sentence, 70% of the internal resources are allocated to separating the training vowels and consonants. Only about 26% of those resources are used to extract more useful information about the sequential structure of the training sentences, such as recognizing the difference between the "A" and "B" words. Hierarchical cluster analysis performed on hidden activation vectors formed at the end of each *test* pattern for 8 different networks shows that there is a high degree of overlap between internal representations formed by the test patterns created with one grammar, and the test patterns created with the other grammar. These results confirm my experimental findings on this network, which show a rather unpredictable and erratic behaviour of the network when handling novel test patterns.

Shultz' (1999) and Shultz & Bale's (2001) models represent two connectionist implementations that successfully mirror the infants' *reported* results. These simulations are performed on a slightly modified cascade-correlation network, called encoder. The models described by Shultz (1999) and Shultz & Bale (2001) are identical, except for the input representation. In the more recent experiment, Shultz & Bale (2001) used a sonority scale, while Shultz (1999) assigned a number between 1 and 8 to each syllable. Shultz & Bale (2001) claim that their results "show that an unstructured neural network model without symbolic rules can simulate infant familiarization and novelty results" (Shultz & Bale, 2001). They also argue that the network exhibits "extrapolative generalization outside the range of the training patterns" (Shultz & Bale, 2001).

According to Marcus, Shultz' model succeeds at this task because it uses "nodes as variables" (Marcus, 2001), and the contents of these variables are copied from one node to another (as shown in Figure 23). In Marcus' view, this means that the model actually implements a classical mechanism (i.e., operations over variables), rather than being an eliminative connectionist structure. However, I have demonstrated that Marcus is mistaken in this argument. In Vilcu & Hadley (2003) we ran an extensive set of experiments very similar to Shultz (1999) and Shultz & Bale's (2001), and found that their model develops connection weights in a much more complicated fashion than in Marcus' conjecture. Although the input representations may indicate that the network could make use of variables on the external layers (e.g., variables *current-word*, *second-word*, and *third-word* of a sentence), I discovered that these "variables" are not simply copied to the network's internal layers. In other words, the network does not manipulate the instances of those variables in a classical algorithmic fashion. I believe there are no

explicitly identifiable "variables" (in Marcus' sense) within this model's internal layers. Therefore, I argue that Shultz (1999) and Shultz & Bale (2001) networks are genuine eliminative connectionist models.

Nevertheless, I also believe that Shultz' (1999) and Shultz & Bale's claims (2001) are substantially overstated. In recent work (Vilcu & Hadley, 2003), we found that even though this model closely mirrors Marcus *et al.*'s (1999) reported data, it has limited generalization capabilities. The network not only has problems extrapolating outside of the training set, but it also has difficulties doing generalization within the range of the training patterns (interpolation). Granted, Shultz (1999) and Shultz & Bale (2001) never explicitly claim their model learns a grammar. However, in saying that the network is able to "recognize a syntactic pattern" (Shultz & Bale, 2001), and has the "ability to learn multiple syntactic forms simultaneously" (Shultz & Bale, 2001), and "[the fact that the networks not only interpolated, but also extrapolated] shows that neural networks are not merely memorizing associations between input and output, but are abstracting functions relating inputs to outputs" (Shultz & Bale, 2001), they imply that their model learns the underlying syntactic structure of the input patterns and is able to successfully apply this knowledge to novel items. The conclusion drawn from our work (Vilcu & Hadley, 2003) is that Shultz' (1999) and Shultz & Bale's (2001) networks behave like a typical pattern recognizer, whose performance is conditioned by *familiar* shapes (numerical contours), rather than like a model capable of discovering *abstract* grammatical relationships.

In accord with the knowledge representation analysis performed on this model, we found that, in general, test sentences closest (in Euclidian space) to the training vectors would generate the smaller network error, regardless of whether those test

sentences had been generated with the familiar or unfamiliar grammar. Therefore, it is Euclidian closeness to the training data, rather than the learning of underlying structure of input patterns, which dictates the behaviour of this model.

Altmann & Dienes (1999) represents another model designed to replicate Marcus *et al.*'s results, and is based on a modified simple recurrent network. One notable characteristic of their model is the partial freezing of weights after training (one set of connection weights are frozen, while another set is able to change even during testing). Another characteristic is the way they measured the network error: by computing the cosine of the angle between the actual and target output vectors. Altmann & Dienes (1999) reported good results for their simulation. They state that they found "significantly higher correlation for congruent sequences than for incongruent ones (...), and a significantly smaller Euclidian distance between prediction and target for congruent targets than for incongruent ones" (Altmann & Dienes, 1999).

However, Marcus has a few objections regarding this model. Firstly, he disagrees with the way Altmann & Dienes interpret the network outputs. He argues that if one uses the most common way to interpret a model, i.e., recording the most active output unit at any time, then the network will not be able to differentiate between the two syntactic structures. A second objection is the way Almann & Dienes do the testing, by freezing some of the connection weights while allowing others to change until the test patterns are learned. Marcus argues that this procedure is unrealistic and "it is unclear what sort of neural system could implement this in the brief period of time which infants have in our experiments" (Marcus, 1999). Finally, Marcus found that Altmann & Dienes' model is not able to correctly generalize to novel sentences, because the model only maps "the

encodings of one set of words onto the encodings of another set of words" (Marcus, 1999).

In Vilcu & Hadley (2003) we duplicated Altmann & Dienes' experiments and discovered serious problems. We found that when the networks were trained with sequences of syllables generated by one grammar, the Euclidian distance between the actual and target vectors was consistently higher for unfamiliar sequences than for familiar sentences, whereas when the networks were trained on the other grammar, the distances were consistently smaller for unfamiliar sentences. We believe these findings are incompatible with the Altmann & Dienes' assertion (1999) that "like the infants (...), our networks successfully discriminated between the test stimuli".

More recently, Altmann (2002) made a few changes to the Altmann & Dienes model and repeated the original simulations. He added a preliminary training phase that supposedly helps the network to "remember" the trained patterns by eliminating the need of generating new internal representations every time a novel input pattern is presented. According to Altmann, the preliminary training phase populates the internal representational space of the network using an extensive set of simple sentences (such as *Noun Verb*, and *Noun Verb Noun*). Another characteristic of Altmann's model is the elimination of the partial freezing of the connection weights during testing. Altmann claims that this model replicates Marcus *et al.*'s results. However, during my own analysis, I discovered that this network is highly dependent on very specific characteristics of the pre-training sentences (i.e., they form in a subtle fashion the underlying ABA and ABB structures), and on the pre-training grammar itself.

The knowledge representation analyses performed on Altmann & Dienes' (1999) and Altmann's (2001) models reveal that these simple recurrent networks do not extract the underlying sequential structure of the input patterns. Principal component analysis shows that most of network's resources (75% in case of Altmann & Dienes, and 70% in Altmann's) are allocated to learning non-useful and non-essential information (with regard to the sequential structure of input stimuli) about the training patterns, such as separating the training sentences based on the middle words. When such a network needs to differentiate between ABA and ABB patterns, it will be unable to separate test sentences that have the same middle word even when those sentences are formed with different grammars. Hierarchical cluster analysis performed on hidden activation vectors formed at the end of each *test* pattern shows a high degree of overlap between grammatical test patterns and ungrammatical test patterns. This confirms my experimental results, which show that the models of Altmann & Dienes (1999) and Altmann (2001) cannot robustly and consistently learn even the simpler grammars.

Christiansen & Curtin (1999), followed by Christiansen *et al.* (2001), employ the same model, namely, a simple recurrent network, initially built for early infant speech segmentation (Christiansen *et al.*, 1998). Christiansen & Curtin (1999), and Christiansen *et al.* (2001) used their existing speech segmentation model in order to replicate Marcus *et al.*'s study on infants. Since Marcus had objections to their earlier work (Christiansen & Curtin, 1999), Christiansen *et al.* (2001) responded with a more comprehensive and detailed experiment. Marcus was initially concerned about the statistical significance of their initial results, since Christiansen & Curtin (1999) only ran their experiments on a single network. Marcus also argued that Christiansen & Curtin's results (1999) were

driven by the network's noise and they could not be replicated. In their most recent work, Christiansen *et al.* (2001) maintain that they replicate their initial experiment on 16 different neural networks and that the results are consistent with regard to their previous study. However, my own analysis of this model shows that Marcus' interpretation is accurate, i.e., the network is mostly driven by noise.

The knowledge representation analysis performed on this model confirms my experimental results. According to the principal component analysis of the hidden activation vectors formed at the end of each training pattern, Christiansen & Curtin's model (Christiansen *et al.*, 2001; Christiansen & Curtin, 1999) learns very particular characteristics of the input stimuli (95% of network's resources are used to do this), such as the difference between various vowels and consonants, as well as the individual letters that are part of the input words. The network is unable to extract the sequential structure of the input stimuli. This indicates that the network will have serious problems with separating the test patterns formed with two different grammars in a robust and consistent way. Indeed, hierarchical cluster analysis performed on hidden activation vectors formed at the end of each *test* pattern shows a high degree of overlap between the two categories of patterns. This finding is consistent with my experimental results, which indicate that there exists an important level of noise in the network.

Negishi (1999) proposed a modified version of a simple recurrent network without hidden units. Each syllable is presented to the network one at a time, and is represented by two input features: the vowel height and the place of articulation of consonants. Negishi reported very good results for his model, and claimed he reproduced the results reported by Marcus *et al.* (1999).

In his analysis of this model, Marcus (2001) argues that it is just an implementation of a classical mechanism, involving two input variables, the network operating in the same way for all instances of those variables. However, I believe that Marcus is mistaken in his interpretation. I have shown that if the two variables that Marcus refers to were place of articulation of consonants (POA) and vowel height (VH), the network could not apply the same operation to all instances of those variables, because such operation was dependent on the context of each instance. Also, if the variables were *current-word* and *previous-word*, my investigation showed that Negishi's model could not learn an abstract relationship between those variables, because the operation that the network learned generated inconsistent results for various test items. Therefore, I argue that Negishi's network does not implement operations over variables, and that it is a genuine eliminative connectionist model, which does simulate the results *reported* by Marcus *et al.* (1999).

Nevertheless, Negishi's model fails to thoroughly learn even the simpler grammars. When minimal changes to the test stimuli (for example, switching the words between the "A" and "B" positions) are made, the model cannot correctly detect the difference between the familiar and unfamiliar test sentences. This experimental result is confirmed by the knowledge representation analysis performed on this model. According to the principal component analysis, most of network's resources (99.4%) are allocated to identifying very specific information regarding the training patterns, such as the various vowels and consonants that occur in the training sentences, or the differences between the numerical representations of vowels and consonants in the first word of each sentence. Once again, this simple recurrent network is unable to extract more useful and important

information from the training patterns (such as abstracting the sequential structure of the input sentences) that could help it differentiate between familiar and unfamiliar *test* stimuli.

9.2.1 Handling of more complex grammars

My analysis of all these connectionist models also contained the investigation of their abilities to perform the same discrimination task when more complex grammars are involved. Based on the Gomez & Gerken's (1999) study on 1-year-old infants, I performed various experiments on all the networks discussed in this thesis. None of these models displayed an ability to differentiate between novel test patterns formed with the familiar grammar, and test patterns formed with an unfamiliar grammar.

9.3 Conclusions

Although some of the eliminative connectionist models discussed in this thesis successfully reproduce Marcus *et al.*'s (1999) *reported* results, they all fail at the more important task of learning the grammars involved. They do not only have problems handling complex (finite-state) grammars, but they also fail with very simple grammars, such as ABA or ABB. The results of both my experimental studies, and those generated by the various analysis techniques employed in this thesis, demonstrate that each of those models is unable to extract the syntactical structure of the input patterns, i.e., they are not capable of discovering the *relationship* among the training stimuli. Instead, these models extract very particular and non-essential features of the training stimuli, such as their numerical contour (Shultz, 1999; Shultz & Bale, 2001), specific characteristics of various vowels or consonants that occur in those stimuli (Christiansen *et al.*, 2001; Christiansen

& Curtin, 1999; Elman, 1999; Negishi, 1999), or uninteresting dependencies on certain words (Altmann & Dienes, 1999).

What is common to all connectionist models that have been discussed in this thesis is the fact that they are trained with the backpropagation algorithm (or a variation of backpropagation called quickprop). As mentioned in the early chapters of this thesis, backpropagation is a learning technique that adjusts the connection weights of a neural network based on the difference (i.e., error) between the target (desired) output and the actual output generated by the network for each training input. During training, the network is taught that, for each input item x_i, $i=1..n$ (where n is the size of the training set), it needs to output y_i, and backpropagation adjusts the connection weights of the network in order to reach that goal. The algorithm is recursive (i.e., the training set may be presented to the network multiple times), and typically stops when it minimizes the difference between target and actual output values (the so-called *error function*) for each input item. In essence, generally speaking, the effect of backpropagation is to find the *simplest* function that minimizes the error function for all x_i. However, since the set of all (x_i, y_i) pairs is just a set of numerical values, the objective of backpropagation is to teach a network how to *numerically approximate* the appropriate output for each input number x_i. Consequently, backpropagation networks implement a numerical function that is purely statistical.

Before analysing the significance of this fact with regard to learning artificial grammars with backpropagation networks, it is noteworthy that, in accord with the results of the various analysis techniques performed in this thesis, backpropagation networks (both feed-forward and simple recurrent networks) are able to discover certain *statistical*

correlations among the *numerical* encodings of the input patterns. For example, Negishi's network discovers that the numerical difference between the encodings (input representations) of consonants and the encodings of vowels contained in the first word of all training sentences are similar. Another example is Shultz's network, which determines that the encodings of the training sentences form certain numerical contours. When presented with novel input, backpropagation networks attempt to numerically approximate that input to a value that is consistent with the network's output to the training patterns. However, this numerical approximation is not always successful. For example, when Negishi's network is tested with novel input that does not match the characteristics of the training patterns (i.e., the numerical difference between the input representations of consonants and vowels of the first word of the novel sentence is different than the numerical differences experienced by the network during training), the network output is inconsistent with the abstract structure of the training patterns (the network error for a novel sentence formed with the unfamiliar grammar is smaller than the network error for a novel sentence formed with the familiar grammar).

Although I disagree with Marcus on several important issues, such as the significance of training independence, the definition of "symbolic" systems, his evaluation of Shultz' (1999), Shultz & Bale's (2001), and Negishi's (1999) models, I agree that backpropagation neural networks have limited generalization capabilities. As mentioned above, I believe that the ability of backpropagation networks to generalize to novel input is dictated by their capacity to numerically approximate a *statistical* function.

Unfortunately for the eliminativists, as argued in (Hadley, 2000), the realm of natural language understanding is not a numerical domain. Although the kind of

statistical functions approximated by backpropagation networks, especially simple recurrent networks, can be fairly complex (Churchland, 1995; Elman, 1990, 1991, 1993), these functions still remain purely statistical. As shown by the various experimental results discussed in this thesis, these numerical, statistical functions appear unable to capture the subtle syntactical and symbolic relationships that are present in natural languages.

Many eliminativists, including Elman, Christiansen, Shultz, Altmann, and others, argue that the kind of statistical functions implemented by backpropagation networks is sufficient in order to explain how humans learn natural languages. In particular, with regard to the ability of backpropagation networks to simulate the language learning processes shown by Marcus *et al.* (1999), Elman maintains that his networks are "inductio engines in which generalizations arise over abstract classes of items" (Elman, 1999), and that "statistical patterns provide the evidence for those classes and for the generalizations over them" (Elman, 1999). On the same subject, Christiansen argues "the statistically-based single-mechanism approach embodied in our connectionist model provides the most simple account of the behavioral data, thus obviating the need for a separate rule-learning component" (Christiansen *et al.*, 2001). Also, Altmann claims that the ability of his simple recurrent network to reproduce Marcus *et al.*'s results "adds to the body of evidence which suggests that models of statistical learning can provide insights into the nature of the conditions that enable certain kinds of learning" (Altmann, 2002).

However, both the experimental results and the analysis techniques performed in this thesis indicate that the statistical mechanisms alone are not sufficient to handle the

kinds of syntactical relationships that occur in Marcus *et al.*'s (1999) and Gomez & Gerken's (1999) training sentences. According to my results, there is no reason to believe that the type of statistical functions approximated by backpropagation networks are powerful enough to explain the symbolic relationships that take place in natural languages. In my opinion, Elman and other eliminativists have not proved yet that statistical functions are adequate to explain natural languages. Until that proof is made, there is no evidence to suggest that the type of backpropagation networks discussed in this thesis can abstract the sequential structures of the input sentences employed by Marcus *et al.* and Gomez & Gerken, and, consequently, are able to consistently generalize to novel input.

Nevertheless, I believe that, in certain conditions, backpropagation networks *can* simulate high-level cognitive processes, such as first order logic, or natural language understanding. These conditions refer to applying certain "classically influenced" (Hadley, 2000) mechanisms to the backpropagation structure, such as handcrafting some or all of the network connections, or employing symbolic AI (Artificial Intelligence) programs. For example, one of the three connectionist models discussed in chapter 8 of this thesis (Implementational Connectionist Models), namely, Shastri & Chang (Shastri, 1999; Shastri & Chang, 1999), employs the backpropagation algorithm and is able to successfully replicate Marcus *et al.*'s study (1999), and freely generalize to novel input. What makes this network succeed is the fact that *it uses a priori knowledge with regard to the structure of the language*, i.e., the network is handcrafted to *explicitly incorporate the grammatical pattern into the networks' structure.*

Shastri & Chang's network makes use of three special input/output units that encode the position of each word within a sentence (the positional role nodes). Significantly, the network does not actually *learn* the positions of words within sentences; this information is explicitly presented to the network. Since the positional role nodes act as output units as well, what the network is taught is to *correlate* the activation of positional role nodes in each training sentence. For example, during ABA training, the network is explicitly taught that the activation of the positional role node P1 correlates with the activation of the positional role node P3. When presented with novel input, regardless of the encoding of the new item, the activation of P1 determines the activation of P3, and vice-versa. Since the network's performance is measured only based on the activations of the positional role nodes, the error will always be smaller for ABA sentences than for ABB sentences.

Although the other two of the three implementational connectionist models discussed in the previous chapter do not employ backpropagation, it is noteworthy that they both are explicitly presented with the sequential structure of the input sentences. For example, Gasser & Colunga (1999) handcraft special relation units, such as sameness and difference units, which are linked to pairs of input units, depending on whether those input units encode similar or different words. In this case, the programmer (the human being that programs the neural network into the computer) transfers his/her a priori knowledge regarding the grammatical structure of the input sentences to the network's structure. The network does not *learn* the grammatical structure, but, rather, the external supervisor explicitly hardwires that grammatical structure into the network. Similarly, Dominey & Ramus' network (2000) is explicitly presented with a numerical encoding of

the abstract structure of the training patterns. This abstract encoding is provided by an external, handcrafted mechanism that employs a typical classical algorithm (it stores up to five input items, and compares the current input with the previous five inputs). This external mechanism (which, in essence, is a symbolic AI program) generates the same numerical encodings for all input sentences that have the same grammatical structure, regardless of whether or not those sentences are "novel". The network is therefore able to freely generalize to any sentence that has the same syntactical structure as the training patterns.

With regard to the debate between classicism and eliminative connectionism, my belief is that eliminative connectionist models in general, and backpropagation networks in particular, have serious difficulties modelling higher-level cognitive processes, such as natural language understanding. This is not to say that classical architectures are automatically the only alternative. Classical systems have their own problems, such as the fragility of their structure, their inability to tolerate noise, and their tendency to only solve domain-specific problems. As argued in (Hadley, 2000), I also believe that neither classicism, nor eliminative connectionism on their own can model all high-level cognitive processes. However, the so-called implementational connectionism looks like a promising paradigm. In addition to the implementational networks discussed in this thesis (Dominey & Ramus, 2000; Gasser & Colunga, 1999; Shastri & Chang, 1999), there are many other studies, such as (Hadley & Cardei, 1999; Hadley & Hayward, 1997; Shastri & Ajjanagadde, 1993), that show the ability of implementational connectionist networks to model many high level cognitive processes.

REFERENCE LIST

Aizawa, K. (1997). Explaining Systematicity. *Mind and Language, 12*, 115-136.

Allen, R., & Reber, A. S. (1980). Very long term memory for tacit knowledge. *Cognition, 8*, 175-185.

Altmann, G. T. M. (2002). Learning and development in neural networks - the importance of prior experience. *Cognition, 85*(2), 43-50.

Altmann, G. T. M., & Dienes, Z. (1999). Rule learning by seven-month-old infants and neural networks. *Science, 284*, 875a.

Altmann, G. T. M., Dienes, Z., & Goode, A. (1995). Modality independence of implicitly learned grammatical knowledge. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 21*, 899-912.

Barone, P., & Joseph, J. P. (1989). Prefrontal cortex and spatial sequencing in the macaque monkey. *Experimental Brain Research, 78*, 447–464.

Brooks, L. R., & Vokey, J. R. (1991). Abstract analogies and abstracted grammars: Comments on Reber (1989) and Mathews et al. (1989). *Journal of Experimental Psychology: General, 120*, 316-323.

Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research, 1*, 245-276.

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

Christiansen, M. H., Allen, J., & Seidenberg, M. S. (1998). Learning to segment using multiple cues: A connectionist model. *Language and Cognitive Processes, 13*, 221-268.

Christiansen, M. H., Conway, C. M., & Curtin, S. L. (2001). *A connectionist single-mechanism account of rule-like behavior in infancy.* Paper presented at the Twenty-Second Annual Conference of the Cognitive Science Society.

Christiansen, M. H., & Curtin, S. L. (1999). *The power of statistical learning: No need for algebraic rules.* Paper presented at the Twenty-First Annual Conference of the Cognitive Science Society.

Churchland, P. M. (1995). *The engine of reason, the seat of the soul: A philosophical journey into the brain.* Cambridge, MA: MIT Press.

Dienes, Z., Altmann, G. T. M., & Gao, S. J. (1999). Mapping across domains without feedback: A neural network model of implicit learning. *Cognitive Science, 23*, 53-82.

Dominey, P. F., & Ramus, F. (2000). Neural network processing of natural language: I. Sensitivity to serial, temporal and abstract structure in the infant. *Language and Cognitive Processes, 15*(1), 87-127.

Elman, J. (1989). *Representation and structure in connectionist models*: Center for Research in Language, University of California at San Diego.

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

Elman, J. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7*, 195-224.

Elman, J. (1993). Learning and development in neural networks: The importance of starting small. *Cognition, 48*, 71-99.

Elman, J. (1998). *Generalization, simple recurrent networks, and the emergence of structure.* Paper presented at the Twentieth Annual Conference of the Cognitive Science Society.

Elman, J. (1999). *Generalization, rules, and neural networks: A simulation of Marcus et al.*, from www.crl.ucsd.edu/~elman/Papers/MVRVsim.html

Elman, J. (2001). *Representational Issues, Commentary on The Algebraic Mind, by Gary Marcus*, from http://www.psych.nyu.edu/gary/elman.html

Fahlman, S. E. (1988). *Faster-Learning Variations on Back-Propagation: An Empirical Study.* Paper presented at the 1988 Connectionist Models Summer School, Los Altos, CA.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems, 2*, 524-532.

Flury, B. (1988). *Common Principal Components and Related Multivariate Models*: New York: Wesley.

Fodor, J. A., & McLaughlin, B. (1990). Connectionism and the Problem of Systematicity: Why Smolensky's Solution Doesn't Work. *Cognition, 35*, 183-204.

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

Gasser, M., & Colunga, E. (1997). *Playpen: Toward an architecture for modeling the development of spatial cognition* (No. 195). Bloomington: Indiana University.

Gasser, M., & Colunga, E. (1999). *Babies, variables, and connectionist networks.* Paper presented at the Twenty-First Annual Conference of the Cognitive Science Society.

Giegerich, H. J. (1992). *English Phonology: An introduction*: Cambridge University Press.

Gomez, R. L., & Gerken, L. A. (1996). *Artificial grammar learning in one-year-olds: Evidence for generalization to new structure.* Paper presented at the Twenty-First Annual Boston University Conference on Language Development.

Gomez, R. L., & Gerken, L. A. (1999). Artificial grammar learning by 1-year-olds leads to specific and abstract knowledge. *Cognition, 70*, 109-135.

Gomez, R. L., Gerken, L. A., & Schvaneveldt, R. W. (2000). The basis of transfer in artificial grammar learning. *Memory & Cognition, 28*(2), 253-263.

Hadley, R. F. (1997). Cognition, Systematicity and Nomic Necessity. *Mind and Language, 12*, 137-153.

Hadley, R. F. (1999). Connectionism and novel combinations of skills: implications for cognitive architecture. *Minds and Machines, 9*, 197-221.

Hadley, R. F. (2000). Cognition and the computational power of connectionist networks. *Connection Science, 12*(2), 95-110.

Hadley, R. F., Arnold, D., & Cardei, V. (1998). Syntactic systematicity arising from semantic predictions in a Hebbian-competitive network. *Connection Science, 13*, 73-94.

Hadley, R. F., & Cardei, V. (1999). Language acquisition from sparse input without error feedback. *Neural Networks, 12*, 217-235.

Hadley, R. F., & Hayward, M. B. (1997). Strong semantic systematicity from Hebbian connectionist learning. *Minds and Machines, 7*, 1-37.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. New York: Wiley.

Hetherington, P. A., & Seidenberg, M. S. (1989). *Is there "catastrophic interference" in connectionist networks?* Paper presented at the Eleventh Annual Conference of the Cognitive Science Society.

Hummel, J. E., & Holyoak, K. J. (1993). Distributing structure over time. *Brain and Behavioral Sciences, 16*, 464.

Jolliffe, I. T. (1986). *Principal Component Analysis*: Springer-Verlag.

MacWhinney, B. (1991). *The CHILDES Project*: Hillsdale, NJ: Lawrence Erlbaum Associates.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive Psychology, 37*(3), 243-282.

Marcus, G. F. (1999). Response to Almann and Dienes. *Science, 284*, 875a.

Marcus, G. F. (2001). *The algebraic mind. Integrating connectionism and cognitive science*: MIT Press.

Marcus, G. F., & Bandi Rao, S. (1999). *Rule-learning in infants? A challenge from connectionism.* Paper presented at the Twenty-fourth annual Boston Univeristy Conference on Language Development, Boston, MA.

Marcus, G. F., Vijayan, S., Bandi Rao, S., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. *Science, 283*, 77-80.

Mathews, R. C., Buss, R., Stanley, W. B., Blanchard-Fields, F., Cho, J. R., & Druhan, B. (1989). Role of implicit and explicit processes in learning from examples: A

synergistic effect. *Journal of Experimental Psychology: Learning, Memory, & Cognition, 15*, 1083-1100.

Matthews, R. (1997). Can Connectionists Explain Systematicity? *Mind and Language, 12*, 154-177.

McClelland, J. L., McNaughton, B. L., & O'Reilly. (1995). Why there are complementary learning systems in the Hippocampus and Neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review, 102*, 419-457.

McCloskey, M., & Cohen, N. J. (1989). *Catastrophic interference in connectionist networks: The sequential learning problem* (Vol. 24): New York: Academic Press.

Meulemans, T., & Van Der Linden, M. (1997). Does the artificial grammar learning paradigm involve acquisition of complex information? *Psychologica Belgica, 37*, 69-88.

Movellan, J. (1990). Contrastive Hebbian learning in the continuous Hopfield model. *Proceedings of the 1990 Connectionist Models Summer School*, 10-17.

Negishi, M. (1999). Do infants learn grammar with algebra or statistics? *Science, 284*, 435.

Newell, A. (1980). Physical symbol systems. *Cognitive Science, 4*, 135-183.

Niklasson, L. F., & van Gelder, T. (1994). On being systematically connectionist. *Mind and Language, 9*, 288-302.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*: MIT Press.

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

Plunkett, K., & Elman, J. L. (1997). *Exercises in Rethinking Innateness: A Handbook for Connectionist Simulations*: MIT Press.

Plunkett, K., & Marchman, V. (1993). From role learning to system building: Acquiring verb morphology in children and connectionist nets. *Cognition, 48*, 21-69.

Reber, A. S. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning & Verbal Behavior, 77*, 317-327.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning internal representations by error propagation*: Cambridge, MA: MIT Press.

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

Seidenberg, M. S., & Elman, J. (1999). Do infants learn grammar with algebra or statistics? *Science, 284*, 435-436.

Sejnowski, T. J., & Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems, 1*, 145-168.

Shastri, L. (1999). Infants learning algebraic rules. *Science, 285*, 1673-1674.

Shastri, L., & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: a connectionist representation of rules, variable and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences, 16*, 417-451.

Shastri, L., & Chang, S. (1999). *A spatiotemporal connectionist model of algebraic rule-learning* (No. TR-99-011). Berkeley: University of California.

Shultz, T. R. (1999). *Rule learning by habituation can be simulated in neural networks.* Paper presented at the Twenty-First Annual Conference of the Cognitive Science Society.

Shultz, T. R., & Bale, A. C. (2001). Neural network simulation of infant familiarization to artificial sentences: rule-like behavior without explicit rules and variables. *Infancy, 2*, 501-536.

Shultz, T. R., & Elman, J. L. (1994). Analyzing cross connected networks. *Advances in Neural Information Processing Systems, 6*, 1117-1124.

Shultz, T. R., Oshima-Takane, Y., & Takane, Y. (1995). Analysis of unstandardized contributions in cross connected networks. *Advances in Neural Information Processing Systems, 7*, 601-608.

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

Vilcu, M., & Hadley, R. F. (2001). *Generalization in simple recurrent networks.* Paper presented at the Twenty-Third Annual Conference of the Cognitive Science Society.

Vilcu, M., & Hadley, R. F. (2003). *Two apparent counter-examples to Marcus: A closer look.* Paper presented at the Twenty-Fifth Annual Conference of the Cognitive Science Society.

von der Malsburg, C. (1981). *The correlation theory of brain function.* (No. 81-2): Department of Neurobiology, Max-Plank-Institut for Biophysical Chemistry.

Wood, C. C. (1978). Variations on a theme by Lashley: Lesion experiments on the neural model of Anderson, Silverstein, Ritz, and Jones. Psychological Review, 85(6), 582-591.