BL DIG

STORM: an Unsupervised Connectionist Model for Language Acquisition

Thomas McQueen

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of Doctor of Philosophy

October 2005

41 0654153 8

ProQuest Number: 10183172

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 10183172

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

321545
THE NOTTINGHAM TRENT UNIVERSITY LIB
Short Loan | PH.D/E1/05
MAC

Abstract

Language acquisition is one of the core problems in artificial intelligence. Current performance bottlenecks in natural language processing (NLP) systems result from a prerequisite for an incalculable amount of language and domain-specific knowledge. Consequently, the creation of an automated language acquisition system would revolutionize the field of NLP. Connectionist models that learn by example (i.e. artificial neural networks) have been successfully applied to many areas of language acquisition. However, the most widely used class of these models, known as supervised connectionist models, have a number of major limitations, including an inability to represent variables and a limited ability to generalize from sparse data. Such limitations have prevented connectionist models from being applied to large-scale language acquisition.

This research considers the alternative and less widely used class of unsupervised connectionist models and investigates whether such models can capture the finite-state properties of language. A novel unsupervised connectionist model, STORM (Spatio Temporal Self-Organizing Recurrent Map), is proposed that uses a memory-rule based approach to learn a regular grammar from a set of positive example sequences. STORM's learning algorithm uses a derivation of functional-equivalence theory that allows the model to learn via *similarity of behaviour*, rather than just *similar of form*. This novel functional generalization ability allows STORM to learn a perfect and stable representation of the Reber grammar from a sparse training set of just 30 sequences, as opposed to the 60,000 sequences required to train a supervised connectionist model. Unlike supervised models, once STORM has learnt the grammar it can generalize to test sequences of any length or depth of embedding.

Extensions to the model are proposed to show how STORM can learn context-free grammars. These extensions also solve the logical problem of language acquisition by recovering from overgeneralizations without the need for negative evidence.

"Connectionism, as a radical restructuring of cognitive theory, will stand or fall depending on its ability to account for human language"

– (Pinker and Prince, 1988).

Acknowledgments

I would like to thank my supervisors, Professor Adrian Hopgood, Dr Tony Allen and Dr Jonathan Tepper for their support and advice throughout this research. The guidance and academic experience provided by Adrian has allowed me to develop as a researcher in AI, while Jonathan's wealth of knowledge in connectional linguistics has helped to shape my perspective of this exciting field. Special thanks must go to Tony Allen whose guidance, rigorous analysis and detailed understanding helped mould a radical theory into a working model. I am also very grateful to Nottingham Trent University for funding my PhD and providing a culture of academic excellence in which to study.

I must also acknowledge the support of my parents, without whom the last three years would have been a struggle, rather than an enjoyable learning experience.

Finally I would like to acknowledge the countless researchers whose work forms the foundations upon which this PhD thesis is built.

Table of contents

Abstract i
Acknowledgments iii
Table of contents iv
List of Figures vi
List of Tables viii
List of Equations xi
1 Introduction 1
1.1 Research objectives 3
1.2 Organization of Thesis 4
1.3 Contribution provided by this research 6
2 – Literature study 8
2.1 Complexity of language with respect to acquisition 8
2.1.1 The formal complexity of language within the Chomsky hierarchy 11
2.2 Connectionist models of language acquisition 16
2.2.1 The case for connectionism 17
2.2.2 Unsupervised connectionist learning algorithms 20
2.2.2.1 Unsupervised connectionist models of language acquisition 24
2.2.2.2 Dynamic unsupervised connectionist models 26
2.2.3 Supervised connectionist learning algorithms 33
2.2.3.1 Supervised connectionist models of language acquisition 34
2.3 Limitations of connectionism 38
2.3.1 Arguments against biological plausibility 38
2.3.2 Arguments against connectionism for developmental cognitive modelling 40
2.3.3 Learning deterministic representations using a continuous state-space 42
2.4 Discussion and conclusions 44
3 – Simple Recurrent SOM (SRSOM) 48
3.1 – Introduction 48
3.2 – Proposed new model 49
3.2.1 - Architecture and algorithm for the SRSOM 50
3.2.2 – Initial experiments to test recurrency mechanism 52
3.2.3 Enhanced Graycode context representation 55
3.3 - Experimental analysis of SRSOM on simple grammars 58
3.3.1 – Connectionist grammar induction 58
3.3.2 – An SRSOM grammar inductor 60
3.3.3 Experiments on the Reber grammar 63
3.3.3.1 Generation of training and test sets 64
3.3.3.2 Experimental parameters 66
3.3.3.3 Experimental results 67
3.3.3.4 Analysis of experimental results 68
4 – STORM 74
4.1 Limitations of the SRSOM – The need for state 74
4.2 Memory-rule based models 77
4.3 STORM (Spatio-Temporal self-Organizing Recurrent Map) 79
4.3.1 STORM's memorization mechanism 79

4.3.1 STORM's rule-based construction mechanism 80
4.3.2 Operation of the temporal Hebbian learning mechanism 83
4.4 Learning via functional generalization 88
4.5 Experiments 90
4.6 Analysis of STORM 97
4.7 Conclusions 102
5 Beyond the Reber grammar 105
5.1 Instability and Causality Loops 105
5.1.1 Causality loops 107
5.2 Recovery from over-generalization 109
5.2.1 Exception construction mechanism 111
5.3 Context-free grammars 116
5.3.1 Sub-sequence rule-construction algorithm 120
5.4 Conclusions 121
6 Conclusions and future work 123
6.1 Future Research 125
6.1.1 Stabilization of model 126
6.1.2 Inflectional morphology, performance evaluation and Mealy machines 127
6.1.4 Experiments on more complex grammars 129
6.1.5 Multi-layered STORM model 130
6.16 Beyond linguistics – The power of functional generalization 132
References 135
Appendix A - Finite-state machine for Reber grammar 142
Appendix B – Glossary 143
Appendix C – Published works 145
Appendix D - Examples of training and test sequences 162

List of Figures

2.1 Kohonen's Self-Organizing Map (SOM) showing the winning neuron for the input pattern (black) and the neighbouring neurons (grey).....21

2.2 Dynamic SOM buffer approach, as used in Kangus's TSOM.....27

2.3 Two identical sequences of ones which have been displaced in time.....28

2.4 Recursive SOM showing how the winning neuron (black) is the best match for the current input and the copy of the map at the previous time step.....30

2.5 U-shaped learning curve of child performance.....35

3.1 Diagram of SRSOM showing feedback of the previous winning neuron's column and row.....50

3.2 Finite state machine (FSM) for the Reber grammar.....59

3.3 Example showing operation of the prediction algorithm.....62

3.4 Training and test sets artificially segregated to proportionally represent all levels up to a specific recursive depth.....65

3.5 Results from 4 test sets for a 5 ×5 SRSOM trained on the Reber grammar....67

3.6 Results from 4 test sets for a 10 ×10 SRSOM trained on the Reber grammar.....68

3.7 Results from 4 test sets for a 20 ×20 SRSOM trained on the Reber grammar.....68

3.8 State diagram showing the winning neurons for each state in the Reber grammar for a 10 × 10 SRSOM.....70

4.1 The winning neurons for two memorized sequences that end with the same sub-sequence 'XSE'.....80

4.2 Illustration of the alternative winning neuron selection algorithm tracing back through the stored sequence BTSXSE.....81

4.3 Functional override in winning-neuron selection algorithm.....82

4.4 Master lateral connection from the 'S' neuron to the 'T' neuron (light arrow) and a slave lateral connection from the 'T' neuron to the 'S' neuron (dotted dark arrow).....84

4.5 Application of negative learning rate to all of neuron 14's lateral connections.....85

4.6 Application of positive lateral learning rate.....86

4.7 Mutual-activation reinforcement.....87

4.8 Results from experiment 1 showing the prediction performance on test set 1 for a 10 ×10 STORM model trained on trained set 1.....91

4.9 Results from experiment 1 showing the prediction performance on test set 2 for a 10 ×10 STORM model trained on training set 2.....92

4.10 Results from experiment 1 showing the prediction performance on test set 3 for a 10 ×10 STORM model trained on training set number 3.....92

4.11 Results from experiment 2 showing the prediction performance on test set 1 for a 5 × 5 STORM model trained on training set 1.....93

4.12 Results from experiment 2 showing the prediction performance on test set 2 for a 5 × 5 STORM model trained on training set 2.....93

4.13 Results from experiment 2 showing the prediction performance on test set 3 for a 5 × 5 STORM model trained on training set 3.....94

4.14 Results from experiment 3 showing the prediction performance on test set 1 for a 15 × 15 STORM model trained on training set 1.....94

4.15 Results from experiment 3 showing the prediction performance on test set 2 for a 15 × 15 STORM model trained on training set 2.....95

4.16 Results from experiment 3 showing the prediction performance on test set 3 for a 15 × 15 STORM model trained on training set 3.....95

4.17 Results from experiment on 15×15 model using training set 1 with random weights, but fixed training sequences presentation order.....96

4.18 State diagram for 100% model.....99

4.19 State diagram for 74% model.....102

5.1 Diagram showing the abrupt context change resulting from the activation of a rule.....106

5.2 Illustration of Causality loop.....108

5.3 Simple regular grammar that contains both rules and exceptions.....110

5.4 Illustration of the over-generalization of an exception to a rule.....111

5.5 Identification of erroneously applied rule.....113

5.6 Flow chart showing revised rule application algorithm to incorporate exception handling mechanism.....114

5.7 Diagram illustration lateral exception connection.....115

5.8 Example of two rules from a context-free grammar.....116

5.9 FSM for Extended Reber grammar.....117

5.10 Diagram showing requirement for context preservation when re-using the sub-sequence TXS from the Extended Reber grammar.....119

5.11 Diagram illustrating use of partial back-trace algorithm to identify alternative winning neurons in sub-sequences.....120

6.1 Multi-layered STORM model.....131

List of Tables

2.1 Production rules for the Reber grammar....12

2.2 Chomsky hierarchy of grammars....13

2.3 Production rules for simple context-free grammar....14

2.4 Different sequences of terminal symbols generated via application of the production rule for non-terminal'A'....14

3.1 The elements and corresponding winning neurons for five sequences which were clustered using the SRSOM operating using a binary context representation....54

3.2 Representation of the winning neurons for the 1st elements of each sequence....55

3.3 Hamming distance for two sets of neurons....56

3.4 Elements and corresponding winning neurons for five sequences clustered using the SRSOM operating with a 2D Graycode context representation....57

3.5 Binary and Graycode representations for two winning neurons....57

3.6 Example sequences generated by the Reber grammar....60

3.7 Orthogonal input vector representation for SRSOM....61

3.8 Symbols and corresponding winning neurons from five selected sequences processed by the SRSOM....72

4.1 Example illustrating the need for state in order to generalize....75

4.2 Example of how the future-context of sequences can be used to identify states in the grammar....78

4.3 Functional Generalization example....89

4.4 Experimental parameters for experiment 1....91

4.5 Activations from a 10×10 model that achieved 100% prediction performance on the Reber grammar....98

4.6 Activations from a 10×10 model that achieved only 74% prediction performance on the grammar....100

5.1 Sequences from the extended Reber grammar illustrating STORMs inability to learn centre-embeddings.....118

List of Equations

2.1 Formal definition of a grammar.....12

2.2 Example of a production rule from a CSG.....15

2.3 Winning neuron selection algorithm for SOM.....21

2.4 Weight update algorithm for SOM.....22

2.5 Neighbourhood function for SOM.....22

2.6 Typical equation for LIN neuron.....29

2.7 Recursive SOM's vector of activities algorithm.....30

3.1 Winning neuron selection algorithm for SOM.....51

3.2 Weight update algorithm for SOM.....51

3.3 Neighbourhood function for SOM.....51

3.4 Algorithm to calculate the distance between two neurons.....51

1 Introduction

The ability to communicate through spoken or written language is considered by many philosophers to be the hallmark of human intelligence (MacWhinney, 2002a). Consequently, since the inception of artificial intelligence, one of its central goals has been to develop systems capable of automatically acquiring and modelling human language. This is particularly appropriate when one considers that most information processing systems either communicate directly with humans or process stored representations of language. However, current natural language processing (NLP) systems possess only elementary abilities to interpret, respond to and ultimately understand human language. For example, Internet search engines typically rely on finding information by simple matching keywords. Such a brute force approach often identifies many irrelevant web pages that happen to contain multiple instances of the keywords, but actually pertain to unrelated topics. For example, searching the web for the keywords *STORM* and *AI* identifies web pages discussing topics ranging from drainage and music to DVDs. Even successful information extraction systems, such as JASPER (Journalist's Assistant for Preparing Earning Reports) (Andersen *et al*, 1992) rely on frame and slot approaches driven by simple pattern matching. Thus despite the growth in information processing, few systems possess the *deep* intelligence necessary to understand natural language.

The main limitation of all current NLP systems is a prerequisite for an incalculable amount of language and domain specific knowledge. The complex and fickle rules of language, which allow people to effortlessly parse recursive sentences and to resolve anaphora, are very hard to manually specify using computational algorithms. Consequently researchers have sought to build models capable of automatically

learning and representing language (Broeder and Murre, 2002). However, designing an automated language acquisition system is a formidable task and despite decades of research a viable solution remains elusive (Andersen et al, 1992).

The problem of language acquisition is considered so complex, that many researchers consider the very concept to be a paradox (Jackendoff, 2002)[1]. Most notably, Chomsky argues that the input to which children are exposed is insufficient for them to determine the grammatical rules of the language (Chomsky, 1965). This argument for the poverty of stimulus theory is based on Gold's theorem (Gold, 1967), which proves that most classes of languages cannot be learnt using only positive evidence. This proof is based on the apparent inability of a learner to recover from the effect of overgeneralization. The argument being that without explicit negative evidence (i.e. a teacher identifying ungrammatical words such as *runned*) learners cannot restrict their model of the grammar to only allow legal sentences.

Gold's analysis and proof regarding the unfeasibility of language acquisition forms a central conceptual pillar of modern linguistics. It is also the basis for Chomsky's theory of *Universal Grammar* (Chomsky, 1965), which proposes that humans have an innate propensity for language. However, recent advances in a discipline known as *connectionism*, also known as artificial neural networks (ANNs), have called the mainstream natavist theories of linguistics into question by demonstrating that biologically inspired learning models can conform to aspects of the human language acquisition process (Rumelart and McClelland, 1986). However, despite the optimism of researchers in this emerging field, connectionist models do have a number of

[1] See (MacWhinney, 2004) for a discussion of the paradox of language acquisition from a connectionist perspective.

serious limitations. These include the inability of connectionist models to represent symbols (Marcus, 1998), a limitation that prevents them from representing abstract relationships such as grammatical states and also a questionable ability to learn from sparse data (Hadley, 1994; Hadley and Cardei, 1999).

1.1 Research objectives

This thesis aims to investigate the capabilities of unsupervised connectionist models in the task of grammar induction. Unsupervised approaches to connectionism have remained in the shadow of the widely studied class of supervised connectionist learning algorithms. One of the reasons for the popularity of supervised algorithms is that such approaches can easily be extended to dynamic models in order to solve temporal problems. While a small number of notable unsupervised dynamic connectionist models have been proposed (Barreto and Araújo, 2001) none of them have been powerful or efficient enough to compete with supervised connectionist models. To date it is believed that no unsupervised connectionist models have been successfully applied to grammar induction. Therefore, the primary objective of this research is to determine the viability of using unsupervised connectionist models for language acquisition. This will be achieved by developing a model capable of inducing a representation of a grammar using only a set of positive examples. The investigation will focus on aspects of connectionist modelling that currently have not been adequately solved using supervised learning algorithms.

1.2 Organization of Thesis

This thesis is structured as follows. Chapter 2 explains the problem of language acquisition and discusses the symbolic and connectionist modelling paradigms. This is followed by a discussion of unsupervised connectionist modelling and leads into a review of the significant unsupervised models in the literature to date. Chapter two concludes with the identification of promising avenues of investigation that may lead towards the research objectives.

Chapter 3 introduces the Simple Recurrent Self-Organizing Map (SRSOM), an unsupervised recurrent connectionist model that extends Kohonen's SOM (Kohonen, 2001) into the temporal domain. The chapter details the design, analysis and refinement of this model. This is followed by experimental analysis of the SRSOM on a simple regular grammar. Chapter 3 concludes with a critical analysis of the SRSOM and the identification of a significant limitation in the model which prevents it from adequately learning the grammar.

Chapter 4 reviews the limitations of the SRSOM and provides an explanation as to why any model seeking to learn the underlying rules of a grammar must form a state-based input representation. The chapter then focuses on memory-rule based models (Pinker, 2000; Marcus, 2000), as a possible method of achieved this required state-based input representation. Chapter 4 presents STORM, a memory-rule based unsupervised connectionist model, based on the SRSOM. The model is experimentally evaluated on the same similar grammar induction problem as the SRSOM. These experiments involve testing various sized STORM models on a number of randomly generated training and test sets, in order to measure the model's

grammar induction abilities. An activation analysis shows that STORM is indeed capable of forming a state-based input representation and learning the underlying rules of a regular grammar. Chapter 4 concludes with a discussion of the model and the experimental results, with respect to the research objectives.

Chapter 5 presents design extensions to the STORM model that explain how the model can be both stabilized and extended to learn more complex grammars. The chapter shows that by optimizing its representation of sub-sequences, STORM can easily be enhanced to learn context-free grammars and recover from overgeneralizations. This is an important aspect of the research and it is shown how the use of an exception handling mechanism will allow STORM to provide an elegant solution to the logical problem of language acquisition. Chapter 5 will also discuss limitations in STORM's learning algorithm that may explain certain instabilities in the model's behaviour.

Finally, chapter 6 reviews the research and evaluates the contribution of STORM, with respect to the original research objectives. The chapter highlights the model's novelty and discusses how it has filled a gap in cognitive modelling by combining the abstract representational power of traditional symbolic approaches with the induction abilities of connectionist models. Future research is proposed that will involve initially stabilizing the model, before implementing the exception handling mechanism discussed in chapter 5. A regime of experiments is proposed to demonstrate how STORM will exhibit the U-shaped learning curve (Brown, 1973) characteristic of human early language learners. The proposal highlights how the model's representations can be modified by linking its winning neurons representing

input symbols to an output alphabet, effectively turning STORM into a Mealy machine (Hopcroft and Ullman, 1979). In doing so this allows meaningful predictions to be made by ensuring that the model's input representations are grounded on more than just abstract symbols.

1.3 Contribution provided by this research

The work presented in this thesis fulfils the research objectives by showing that an unsupervised connectionist model can be successfully applied to the problem of language acquisition. With respect to this problem, the research challenges traditional natavist theories of linguistics by showing that the rules of a grammar can be learnt using only positive examples and a sparse training set. More importantly, a design based on the STORM model is used to explain why negative evidence is unnecessary in order to account for recovery from overgeneralization in a memory-rule based model. This poses a possible answer to the age-old question, formalized by Gold's theorem, of how a language learner can recover from overgeneralization without the need for explicit negative evidence.

Outside the context of linguistics, STORM is a major contribution to connectionism in itself. By proving that an unsupervised connectionist model can successfully be used in temporal sequence processing, this research may help diversify the field by drawing unsupervised learning algorithms out of the shadow of their supervised counterparts. STORM's unsupervised learning mechanisms make it more biologically plausible than equivalent supervised models and may therefore help to counter some of the key arguments (Jackendoff, 2002) against connectionist approaches to cognitive modelling. STORM's induction and functional-generalization abilities may

potentially reveal interesting structures and relationships in many complex dynamic problems, such as financial forecasting (Yao and Tan, 2001) and robot planning and control (Platt *et al* 2003).

In the context of memory-rule based models and dual-based learning mechanisms, STORM's rule based approach to learning provides a domain independent, general purpose approach to rule construction. While memory-rule based models have been previously proposed (Marcus, 2000), such models have either been theoretical or used hard-wired rules. STORM's criterion of using regularities in the future-context to construct rules therefore provides an answer to the advocates of memory-rule based models, who have long searched for a mechanism by which to learn rules.

2 – Literature study

This chapter begins with a discussion of the complexity of the language acquisition process. This details the paradox of language acquisition and explains the mainstream linguistic perspective regarding the unfeasibility of this acquisition process.

The complexity of the computational mechanisms involved in the language acquisition process is described in relation to the Chomsky hierarchy (Chomsky, 1959). This is discussed along with pertinent terminology from automata theory (Hopcroft and Ullman, 1979). The main body of this chapter discusses connectionism, focusing on its biological basis along with its applicability to modelling language acquisition. The two main connectionist learning paradigms of supervised and unsupervised learning are discussed, along with notable models of language acquisition for each respective paradigm. The limitations of connectionism are discussed, focusing on the shortcomings of the popular supervised connectionist learning paradigm. The chapter concludes by highlighting the potential of unsupervised connectionist models with respect to language acquisition.

2.1 Complexity of language with respect to acquisition

The apparent ease with which children acquire language is testimony to the tremendous power of the human mind. This ability to communicate through either spoken or written language is considered by many philosophers to be the hallmark of human intelligence. The general term 'linguistics', used to define the study of language, encapsulates many areas of study including syntax, semantics and pragmatics (Jackendoff, 2002). It also draws on many disciplines including psychology, neuro-biology, sociology and computer science. One of the original and

arguably most important fields of linguistic research involves the study of syntax. Despite the varying theories regarding the syntactic structure of language (Chomsky, 1981; Pollard and Sag 1994), the objective of research in this field has always been to model the rules which constrain the regularities in language. These regularities, such as the '-ed' ending for regular verbs and the gender of pronouns in particular contexts, can be found throughout all levels of language. Entire textbooks are dedicated to rules and exceptions describing the regularities in just small sub-domains of the English language, such as inflectional morphology (Stump, 2001).

While formal linguistic theories differ in their views regarding the acquisition of the rules governing the structure of language, a major objective of linguistic research involves understanding and describing such rules. The development of an automated language acquisition system capable of learning the rules and structure of language on a large scale would be viewed by many researchers as the holy-grail. Such a model would allow the large scale induction and manipulation of knowledge directly from the countless existing electronic databases and other human-readable media. Consequently, improvements in the ability to model and therefore understand natural language, would impact on all applications of AI, from translation (Arnold *et al,* 1993) and natural language understanding (Allen, 1995) to speech recognition (Beaufays *et al,* 2001).

Language was described by the philosopher Wilhelm von Humboldt as *the infinite use of finite media* (Humboldt, 1836/1972). The power to combine known words, within the constraints of syntax and semantics, allows for the expression of ideas in a potentially infinite number of different forms. Due to this combinatorial power of

natural language sentences with similar meanings can be expressed in a potentially infinite number of different ways. If a model could learn a representation of even the derivational rules of the English language (assuming such acquisition is possible in isolation), then it would make it significantly easier for a natural language processing system to parse sentences. For example, in an NLP security application it would be highly desirable to automatically identify the similar meaning in the following two sentences; "The diplomat will be targeted by a suicide bomber hidden in the crowd and assassinated at precisely six o'clock", "At exactly six o'clock the suicide bomber, who will be hidden somewhere in the crowd, will assassinate the diplomat". Thus if a language acquisition model could learn the rules governing the transformation of sentences, it would enable an NLP system to easily identify the underlying form of all sentences.

Many researchers consider the problem of language acquisition to be a paradox. Advocates of the poverty of stimulus theory argue that the fragmentary evidence available to language learners is too inconsistent and incomplete to allow induction of language without some innate predisposition towards the acquisition of language. This creates a paradox, because children who are raised in social environments are almost always able to learn their native language, while those who are isolated from linguistic input as children do not ever acquire a proper language as adults (Jackendoff, 2002). Therefore, if the linguistic input that children are exposed to really is insufficient then why do children who are exposed to this input learn language? The poverty of stimulus theory is supported by Gold's theorem, which proves (under strict assumptions) that an infinite grammar cannot be learnt using only positive examples due to the problem of overgeneralization. This theorem and its

implications for language acquisition, form the central conceptual pillar of modern linguistics. As such, Gold's theorem is also the focus of much debate among those who view the process of language acquisition as a learning process (MacWhinney, 2004).

Gold's theorem and the resulting paradox of language acquisition formed the basis of Chomsky's theory of universal grammar. This controversial and widely misinterpreted theory is concerned with the human innate pre-specification to language acquisition. However, rather than stipulating that all aspects of language from the lexicon to the grammatical rules are innate (as its name may suggest), the theory of universal grammar stipulates that the brain's language acquisition capacity is innate. Universal grammar attempts to sidestep the poverty of stimulus theory by proposing that language learners are born with a functional pre-specification to grammar acquisition. This innate knowledge limits the form of an acquired grammar to that of possible human languages and also contains a strategy by which to select a grammar compatible with the linguistic input.

2.1.1 The formal complexity of language within the Chomsky hierarchy

The study of language within the discipline of computer science (computational linguistics) led to the creation of a formal modeling technique known as generative grammars (Chomsky, 1959). By formalizing a set of production rules, such grammars provide a model that describes all the legal sentences in a language.

Equation 2.1 shows that a grammar is composed of four distinct elements; sets of terminals, non-terminals, production rules and a start symbol. Terminal symbols in a

grammar consist of elements that cannot be broken down into sub-elements (i.e. words such as "man" or "walk"). Non-terminals however can be broken down and are used to represent phrases and parts of speech (i.e. nouns or verbs). The production rules in a grammar specify which non-terminal symbols can be re-written into which terminal symbols.

$$G = \{\Sigma, N, P, S\}$$

Equation 2.1 – Formal definition of a grammar. The ∑ symbol is a set of terminal symbols, N is a set of non-terminal symbols, P is a set of production rules and S is the start symbol.

In table 2.1 the production rules for the Reber grammar (Reber, 1967) show how a set of non-terminal symbols can be replaced with terminal symbols to generate a potentially infinite number of unique sentences. The grammar is represented graphically by the finite state diagram in pullout appendix A. The Reber grammar, which is also used in the experiments in chapter 4, was chosen because it was originally used to investigate implicit rule-learning in human subjects (Reber, 1967).

Non-terminals	Productions rules
S	S1 S2 S4 S6
S	S1 S3 S5 S6
S1	*T* S2
S1	*P* S3
S2	*S* S2
S2	*X* S4
S3	*T* S3
S3	*V* S5
S4	*X* S3
S4	*S* S6
S5	*P* S4
S5	*V* S6
S6	*E*

Table 2.1 - Production rules for the Reber grammar. Each non-terminal symbol (left column) can be replaced by the set of terminal and non-terminal symbols in the corresponding production rule (right column).

The Chomsky hierarchy (table 2.2) shows that generative grammars can be categorized into four types according to their complexity. As shown in table 2.2, the simplest type of grammar is the class of regular grammars (which include the Reber grammar). This simple type of grammar is characterized by the single non-terminal symbol on the left hand side of the production rule and the limit of one terminal symbol on the right. The Reber grammar is known as a memory-less grammar because the next valid states can always be predicted given the current grammatical state (i.e. knowledge of previous symbols in the sequence is not needed).

Grammar type	Name	Alias
0	Recursively enumerable grammars	Unrestricted phrase structure grammars
1	Context sensitive grammars (CSG)	
2	Context free grammars (CFG)	Push down automata
3	Regular grammars	Finite-state grammars (FSG), deterministic finite automata (DFA)

Table 2.2 – Chomsky hierarchy of grammars.

Context-free grammars (CFGs) occupy the next level up the hierarchy from regular grammars. They are characterized by their production rule that allows a non-terminal to be replaced by a set of any number of terminals and non-terminals, including the same non-terminal that is being replaced. Because context-free grammars allow for the replacement of non-terminal symbols with more than one terminal and non-terminal, any computational mechanism capable of processing context-free languages must incorporate a memory. This requirement arises because allowing non-terminal symbols to occur within the right hand side of other production rules, introduces an embedding.

Non-terminals	Production rules
S	A
A	*z* B *z*
A	*y* B *y*
B	*x* C *v*
C	*u* A *i*
C	*k*

Table 2.3 – Production rules for simple context-free grammar.

As shown in table 2.3, the non-terminal 'A' can be replaced by the terminal 'z', followed by the non-terminal 'B' and a terminal 'z'. However, because the non-terminal 'B' can itself be replaced by other terminals and non-terminals (including the original non-terminal 'A'), the first terminal 'z' may be separated from the last terminal 'z' by potentially any number of symbols (table 2.4). Therefore, when processing sentences generated using non-terminal 'A', the prediction of the next symbol depends not just on the current state, but also upon previous symbols. For example in sequences 1 and 2 in table 2.4 the prediction of whether the last symbol is a 'z' or a 'y' depends upon which symbol occurred first in each sequence. This dependency becomes more complex as the level of embedding increases, as shown in sequence 3.

#	Sequence of terminal symbols
1	*z x k v z*
2	*y x k v y*
3	*z x u y x k v y i v z*

Table 2.4 – Different sequences of terminal symbols generated via application of the production rule for non-terminal 'A' (table 2.3). The choice of the second terminal symbol in the production rule for non-terminal 'A' can be determined by the first terminal symbol.

The name *context-free* derives from the freedom this grammar provides to replace the left hand side of a production rule with the right hand side, regardless of the context the left hand side appears in. CFGs are widely used in NLP programs and most parsers treat English as a context free language (Allen, 1995).

Level one of the Chomsky hierarchy is occupied by context-sensitive grammars. As their name suggests these grammars allow restrictions on the replacement of the left hand side of the production rule. This allows a production rule to be applied only in context-specific circumstances (equation 2.2).

$$xAy \Rightarrow xabcdy$$

Equation 2.2 – Example of a production rule from a CSG. The non-terminal 'A' is replaced by the set of terminals 'abcd' only where a terminal 'x' is followed by a terminal 'y'.

The top level of the Chomsky hierarchy is occupied by recursively-enumerable grammars. They are also called unrestricted phrase structure grammars because either side of their production rules can contain any sequences of terminals and non-terminals. These powerful grammars are believed to be a close approximation of natural language (Chomsky, 1959) and their use is unique to human beings (Fitch and Hauser, 2004). Research has shown that monkeys, whom are able to learn simple regular grammars, appear incapable of mastering the rules found in unrestricted phrase structure grammars (Fitch and Hauser, 2004).

2.2 Connectionist models of language acquisition

Traditional symbolic linguistic models sidestep the process of language acquisition by focusing on describing linguistic performance using sets of rules and exceptions. Such a top-down approach to cognition, attempts to work backwards from formal linguistic structure towards human processing mechanisms. While symbolic approaches are very powerful, the resultant models are usually inflexible and cannot easily be applied to general purpose linguistic problems (Corrigan and Iverson, 1994). This inflexibility arises from their hardwired rules and exceptions which are stipulated by the system's designer, rather than learnt by the model itself. Consequently such systems require new sets of rules and exceptions when applied to different problems.

In the past fifteen years an alternative approach to cognitive modeling has once again gained popularity among researchers. Known as connectionism, this empirical field of study uses models whose design is biologically inspired by the operation of the human brain and nervous system. In contrast with symbolic models of cognition, connectionism uses a bottom up approach to cognition that models the learning process itself. Connectionist models are well suited to the problem of language acquisition because they learn the solution to a problem via a set of examples.

Connectionist models also provide a means to overcome the theoretical limitations on language acquisition imposed by Gold's theorem. While Gold proved that an infinite grammar could not be learnt from only positive examples, this proof was based on the assumption that successful acquisition would result in a deterministic grammar. Therefore acquisition could only be said to be successful if the language learner possessed a perfect representation of the grammar and therefore never made any

mistakes regarding its use. (Horning, 1969) exploited this unrealistic assumption to show that language can be learnt from only positive examples if the language identification criterion uses a stochastic probability of success. This stochastic, as opposed to deterministic, view of grammar induction is supported by much language acquisition research, including U-shaped learning curves. Such research suggests that learning proceeds through stages, in which linguistic proficiency increases. However, it also shows that how ever proficient native speakers become, they still occasionally make grammatical errors. Thus no one possesses, or is able to employ, a perfect knowledge of grammar.

2.2.1 The case for connectionism

A large body of evidence exists to support connectionist models of cognition. This evidence comes from neuro-biological and psycho-linguistic research conducted over the last half of this century. Such evidence disputes the views of traditional researchers such as Fodor (1983), who argue that language, along with many other higher cognitive functions, must be innate. A primary natavist argument is that cognitive functions such as language exist in modules (Fodor, 1983). It is suggested that the cognitive micro-circuitry of these modules is genetically pre-specified to perform particular functions.

Over the past two decades the science of genetics has evolved dramatically, cumulating with the complete mapping of the human genome in 2001 (Venter *et al*, 2001). Modern geneticists view the genome of regulatory organisms, such as human beings, as a recipe of ingredients, as opposed to a specific blueprint. Individual genes rarely control specific traits such as eye colour, but rather work along with other

genes and can be involved in multiple functions. Indeed, such a coding strategy is necessary given it is believed that the human genome has only around 25,000 protein-coding genes (Venter *et al*, 2001), while the human body itself is made up of at least 100 trillion cells. Furthermore, humans share over 98% of their DNA with chimpanzees (Elman *et al*, 2001). Given that it is widely established that only humans posses the expressive power of language (Chomsky, 1972), any innate specification for a pre-programmed language module would have to share less than 2% of the human genome, along with every other characteristic distinguishing humans from chimpanzees. This representational limitation of the human genome calls into question innate theories involving pre-specified language, due to the huge burden it would impose on the genome. Furthermore, the complexity of gene interactions and the lack of a genetic blueprint cast serious doubt on nativist theories which rely heavily on pre-specified knowledge and rules, due to issues of representation.

A further counter-argument against innately pre-specified cognitive modules regards recent neuro-biological evidence concerning cortical plasticity (Elman *et al*, 2001). Lessoning experiments on animals (Webster *et al*, 1995) have shown that when the usual area responsible for a specific cortical function is removed in infancy, alternative areas take over. However, when the same area is removed in adulthood, the lessoned animals never recover the associated cognitive functions. Related experiments support the view of cortical plasticity by showing that when inputs from the visual cortex are rewired to the auditory cortex, similar organization is observed as to that which occurs in the visual cortex (Roe *et al*, 1992). Research into cortical activity during language processing tasks shows that the organization of cells

involved in language use varies widely between individuals (Damasio and Damasio, 1992), thus further weakening the argument for innate modules.

The cognitive plausibility of connectionist models has received evidential support based on studies of brain damaged subjects. In cognitive science, the study of brain damaged individual's performance on specific tasks has been very helpful in understanding the cognitive mechanisms involved (McLeod *et al*, 1998). Linguists have experimented on brain damaged subjects in order to gain an understanding of the mechanisms involved in language (Bates *et al*, 1997). However, traditional symbolic models of cognitive processes have found it very hard to account for the types of performance errors caused by brain damage (McLeod *et al*, 1998). In contrast many connectionist models have been proposed to account for brain damage (Hinton and Shallice, 1991; Farah and McClelland, 1991; Cohen and Servan-Schreiber, 1992). Hinton and Shallice (1991) used a connectionist model that mapped visual representations of words onto their associated semantic representations. By lessoning trained models they produced similar visual-semantic errors to that of dyslexics (i.e. the word *night* produces the semantic representation for *sleep*). Their analysis of this model proposes that the performance errors observed in human brain-damaged subjects can best be explained by a distributed attractor based model.

Connectionist models can be broadly categorized into two distinct types based on the principles that govern their learning algorithms. The following two sections review supervised and unsupervised approaches to language acquisition, highlighting significant models in each respective paradigm.

2.2.2 Unsupervised connectionist learning algorithms

Unsupervised connectionist learning is an approach to cognitive modelling whose operating principles remain close to those of biological neural networks. Unlike their supervised counterparts, unsupervised models don't require an external teacher signal that stipulates a desired output or behaviour. The majority of unsupervised models are derived directly from Hebb's law (1949) and self-organize in response to external input stimuli. Hebb showed that learning occurs via the correlation of activity between neurons (i.e. nodes that fire together, wire together). In an unsupervised auto-associator (McLeod *et al*, 1998) this may take the form of strengthening the synapses between neurons[2] representing an input pattern and the corresponding output neurons representing that same pattern. Whereas in an unsupervised competitive model, learning occurs by strengthening the synapses between neurons representing the input pattern and the neuron that is the best match for that input pattern. Unsupervised learning is more biologically plausible than its supervised counterpart because it uses locally available information, between a neuron's axon and dendrite, to update the weights.

Unsupervised learning doesn't require any external target signal as it is driven purely by the principles of self-organization. The organization of an unsupervised neural network is governed by the relationships that exist within the data being processed, rather than by forcing a relationship between the data and some predefined target domain. The dominant connectionist paradigm of supervised learning concentrates solely on how the data can be used to solve a specific problem. In doing so it makes the assumption that any relevant relationships that hold amongst the data will reveal

[2] The term *neuron* is used to refer artificial and biological neurons interchangeably.

themselves in the course of solving the problem. In contrast, unsupervised learning takes the opposite approach, concentrating instead on relationships in the structure of the data itself and potentially revealing internal relationships that may be key to solving the problem at hand.

The most common unsupervised connectionist model is Kohonen's self-organizing map (SOM). As shown in fig 2.1, this model usually consists of a rectangular grid of neurons, each of which has weights connecting it to a layer of input neurons. This model mimics the competitive approach to learning that is believed to operate throughout the central nervous system and brain.

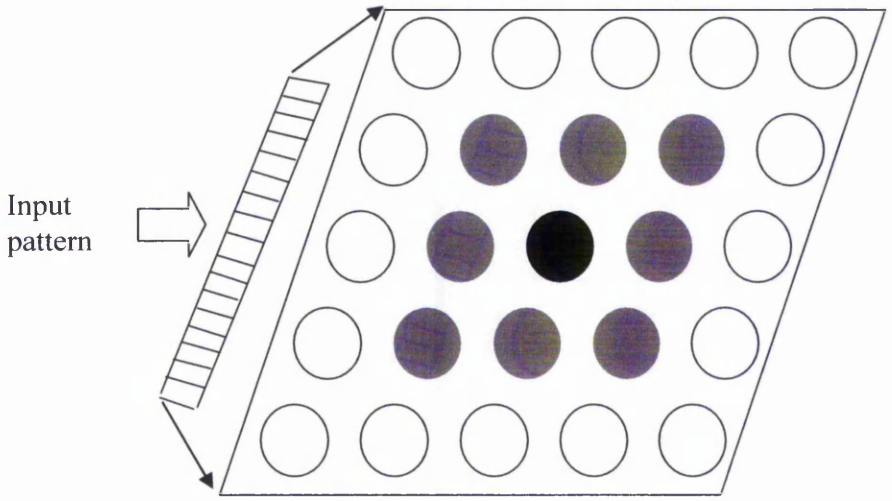


Fig 2.1 –Kohonen's Self-Organizing Map (SOM) showing the winning neuron for the input pattern (black) and the neighbouring neurons (grey).

In competitive learning, a group of neurons compete for the same input signal. The particular neuron which produces the highest activation, with respect to the input signal, is deemed the winner (equation 2.3).

$$N = \arg(\min_j(\sum_i |X_i - W_{ij}|)) \qquad (2.3)$$

The winning neuron and perhaps also its close neighbours participate in learning. This process involves modifying a neuron's synaptic connections in order to make it a better match for to the input in question (equation 2.4). This has the effect of clustering similar inputs together on the map. α is the learning rate coefficient and is typically 0.1 or less. The learning rate coefficient is decreased, usually linearly, throughout training.

$$W_{ij}(t+1)=W_{j}(t)+\alpha h_{ij}(X(t)-W_{ij}(t)) \qquad (2.4)$$

A neighbourhood function h is used which has the effect of moving the weight vectors of neurons adjacent to the winner closer to the input vector than those of neurons further away from the winner. A typical function used to define the neighbourhood is the Gaussian function shown in equation 2.5. This function uses the distance d between the winning neuron and the neuron in question to calculate the influence of the neighbourhood function so that it is proportional to the distance between the two neurons. The parameter σ represents the width of the Gaussian function (in neurons) and controls how many neurons participate in training.

$$h_{ij}=\exp(\frac{-d_{ij}^{2}}{2\sigma^{2}}) \qquad (2.5)$$

During training, the neighbourhood is reduced from a value covering approximately half the number of neurons in the map down to a value effecting one neuron. The SOM is trained with input vectors, presented in random order, either for a preset number of epochs or until the network has converged (i.e. when little or no further weight changes occur). The average quantization error may be used as a measure of convergence and involves measuring the distance between an input vector and the weight vector of the winning neuron. However, most SOM models are trained for a

preset number of epochs. Kohonen (2001) recommends that this training should be carried out in two phases; a convergence phase and a fine-tuning phase. In the convergence phase, the SOM is trained with a relatively high learning rate of 0.1 for 1000 epochs. Both the learning rate and the neighbourhood are decreased throughout training. The fine-tuning phase employs a fixed learning rate of 0.01 and a fixed neighbourhood of 1 is used. The number of epochs in the second phase should ideally be 500 times the number of neurons in the model (Kohonen, 2001).

The competitive approach to learning used by the SOM is grounded in a large body of neuro-biological research which shows that a topological organization is preserved between the central nervous system and the receptors on sensory organs (Kohonen, 2001). For example, a tonopopic map can be found in the auditory cortex in which the organization of cells reflects the pitch and frequency of tones received by the listener's ear (Kohonen, 2001). However, these order-preserving maps are not limited to representing spatial relationships. Neural-imaging research has shown that more abstract, geographical maps can be observed in the hippocampus of rats (Olton, 1977, cited in Kohonen, 2001). Unlike their supervised counterparts, unsupervised models make no prior assumptions about classes of data being clustered. This makes them very powerful for exploring data where the featural characteristics are unknown or where manual construction of input and target vectors is not possible. SOMs are also used as a visualization tool, due to their characteristic ability to map relationships in the data onto a 2D grid. SOMs have been applied in various real world problems, including process monitoring (Kasslin *et al*, 1992), pattern recognition (Kohonen, 2001) and robot arm control (Ritter *et al*, 1992).

Despite the compelling neuro-biological evidence in support of competitive learning and the success of models such as the SOM, far less research has been carried out into the field of unsupervised connectionism than its supervised counterpart.

2.2.2.1 Unsupervised connectionist models of language acquisition

The numbers of unsupervised connectionist models that have been applied to language acquisition are eclipsed by the number of equivalent supervised models that dominate the literature. However, there are a few significant unsupervised models that are relevant to this research. As with other applications in unsupervised connectionism, the majority of language acquisition models are based on the SOM. In (Ritter and Kohonen, 1989) the SOM was applied to producing context-maps. By clustering representations of input words along with both the immediately preceding and following words, such models are able to capture the statistical occurrences of words in specific contexts. This allows the formation of so called *context maps*, in which nouns and verbs are mapped into different regions of the map.

Notable research into context maps includes experiments involving the clustering of natural text taken from Usenet newsgroups on the Internet (Kohonen, 2001). Results from these experiments showed that context maps were very successful at producing closely clustered categories of related words (an effect that is highly desirable in areas of NLP such as data mining (Craven and Shavlik, 1997)). Context maps have also been used as a tool for visualizing the semantics of words formed using a collection of text from Grimm fairy tales (Honkela *et al,* 1995). Larger scale experiments have also been performed by Langus and Kohonen (Lagus *et al*, 1999) that involved visualizing entire document collections using the WEBSOM model.

Context maps operate by producing clusters of words based on the statistical frequency of their usage. They effectively rely on surface similarities and make no attempt to model the underlying structure of language. In order to perform larger scale language acquisition tasks such as grammar induction or parsing, a context map would need to represent all relevant input words in a sentence simultaneously (i.e. a buffer). This would result in *the problem of 2* (Jackendoff, 2002) (i.e. how to internally represent multiple instances of the same word) and also the general problem of how to determine the length of the buffer. Therefore, due to their static nature, context maps may be inappropriate for many language acquisition tasks.

MacWhinney (2002b) employed a hybrid SOM based model in the task of lexical acquisition. The model was proposed as a means of overcoming computational problems encountered in distributed supervised lexical acquisition models. The task of acquiring words is treated as a process of associating phonological features with the respective semantic representations. MacWhinney's model achieved this by used two SOMs and a Hebbian learning mechanism (Hebb, 1949). The first SOM clustered the phonological inputs, forming an auditory map, while the second SOM organized itself using the semantic inputs. The Hebbian learning mechanism was then used to associate the auditory forms of a word in one map, with that word's semantic representation in the other map. The model was able to successfully learn 6,000 sound-meaning patterns, therefore significantly superceding the capabilities of equivalent distributed models. The model was also successfully applied to the problem of learning inflectional morphology (MacWhinney, 2002b).

The power of MacWhinney's model is derived from its localist architecture, which provides stability by representing lexical items in a discrete manner. This allows the model to overcome many of the problems associated with purely distributed representations and scale up closer to natural language. However, the model is not designed to process dynamic sequences and therefore, in its current form, it would not be suitable for language acquisition tasks such as grammar induction.

Another notable unsupervised model is Hadley's semantic parser (Hadley and Cardei, 1999). This model uses both self-organizing and Hebbian based learning to produce semantic parses for both simple and complex sentences containing active and passive verbs. Representations of lexical items presented at the input layer are fed into a SOM, along with inputs from a feature layer. The activations from the SOM form input to the output layer, along with direct connections from the input layer. This output layer contains nodes which represent concepts and roles (ex. "love", "cats", "see" and "mice"). During training the model learns associations between sentences presented at the input layer and its corresponding meaning, which is presented to the output layer. While technically speaking this is a supervised model, its use of self-organization and Hebbian learning demonstrate that the mechanisms used in unsupervised learning algorithms can be successfully employed in the language acquisition process.

2.2.2.2 Dynamic unsupervised connectionist models

While relatively few unsupervised connectionist models have been applied to the problem of language acquisition, numerous such models have been applied to other similar problems involving temporal sequence processing (TSP) (Barreto and Araújo,

2001). These models typically extend the SOM into the temporal domain using various short term memory mechanisms. Given the relation between TSP and language acquisition, these dynamic unsupervised connectionist models are highly relevant to the subject of this thesis. Therefore, this section will briefly review the significant dynamic connectionist models in the literature.

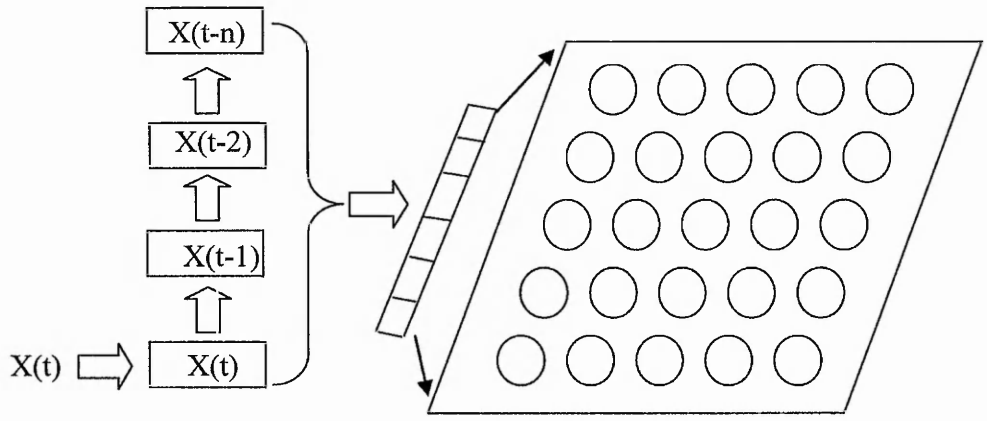


Fig 2.2 – Dynamic SOM buffer approach, as used in Kangus's TSOM.

Kangas's TSOM (1990) was an early dynamic extension to the SOM which used a fixed size buffer that allowed it to cluster sets of inputs, as opposed to individual input clustering. This buffer or sliding window approach (fig 2.2) is the simplest type of short-term memory mechanism. It involves the use of a buffer containing the *n* most recent inputs. The buffer in fig 2.2 is treated like a shift-register, with new inputs being added at the right, while inputs already within the buffer are shifted to the left. This approach essentially maps time onto space, converting a temporal sequence into a spatial pattern. The neural network can then process this pattern in the same manner as any other spatial pattern.

The major limitation of the temporal window approach is that the choice of window size is problem dependent and therefore needs to be selected via a process of trial and error. If the window is too big, then as well as being computationally intensive the model may be swamped with irrelevant data. Conversely, if the window is too small then critical past inputs may be missing, preventing the model from making a correct prediction / classification. Another fundamental problem with a fixed buffer approach is the inability to distinguish a relative temporal pattern from an absolute temporal pattern (Elman, 1990). For example, if the model uses a buffer of eight inputs then the two sequences shown in fig 2.3 will be treated separately.

[0 0 0 1 1 1 0 0]

[0 1 1 1 0 0 0 0]

Fig 2.3 – Two identical sequences of ones (111) which have been displaced in time.

Another notable dynamic SOM which employed buffers is Kohonen's Hypermap (Kohonen, 1991). This model uses multiple buffers, each of which centres on a specific input and holds contextual information at an increasingly higher level of abstraction. The model uses the highest level buffer to identify the general region on the map and then uses lower level buffers to narrow down the winner to a specific neuron. While the Hypermap is a very powerful model, it is computationally intensive due to its use of multiple buffers and corresponding weights. This computational intensity would be compounded as the number of neurons in the map is increased. Additionally, despite using multiple buffers the Hypermap is still subject to the problem of distinguishing between absolute and relative temporal patterns.

In order to overcome the limitations posed by buffers, further dynamic extensions to the SOM employ *leaky integrator neurons* (LINs) (Barreto and Araújo, 2001). Also known as dynamic neurons, LINs maintain a potential, akin to the membrane activity in a biological neuron. Their current potential is integrated with the input to produce the neuron's output value, which also becomes the new potential. The integration function has the effect of decaying information about past inputs and integrating this with information about the current input. A typical equation for a LIN neuron is shown in equation. 2.6.

$$P_i(t) = \lambda P_i(t-1) + X_i(t) \qquad (2.6)$$

Where $P_i(t)$ is the potential of node i at time t, $X_i(t)$ is the input value for node i at time t, λ is the decay coefficient and $P_i(t-1)$ is the potential of node i at the previous time step. Chappell and Taylor's Temporal Kohonen Map (TKM) (1993) and Varsta's Recurrent SOM (RSOM) (Varsta *et al.*, 1997) both use LINs to capture the SOM's outputs and inputs, respectively.

The major advantage of LINs over buffers is that inputs can be presented to the network one at a time as they appear in the sequence. This both eliminates the problems associated with the window size and also reduces the number of trainable weights needed. LIN's have a potentially infinite temporal depth i.e. without noise they can represent information about potentially all elements in a sequence. However, the disadvantage of LIN's are that they have a limited temporal resolution in that they hold increasingly less information about past inputs. This limited resolution can make it virtually impossible for the predictor / classifier to extract sufficient information about past inputs even without the presence of noise (Mozer, 1993).

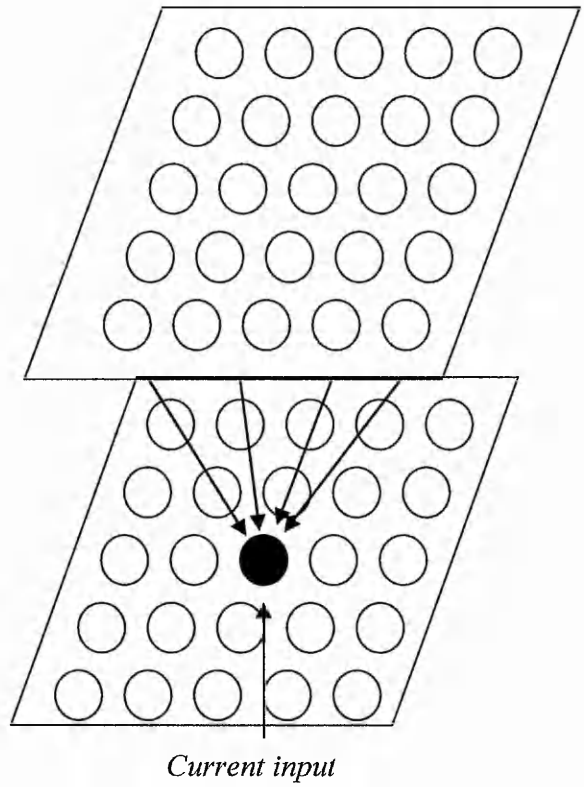


Figure 2.4 – Recursive SOM showing how the winning neuron (black) is the best match for the current input and the copy of the map at the previous time step.

The Recursive SOM (Voegtlin and Dominey, 2002) (fig 2.4) is a dynamic extension to the SOM that uses recurrent connections to propagate the output activity of all its neurons back as a factor of the next input. The winning neuron is the unit whose weight vector best matches the current input vector and a vector of activities from the previous time step. The Recursive SOM's vector of activities is a representation of the contextual information from the previous time step and consists of a representation of the activity of every neuron. Each neuron's activity is calculated using equation 2.7.

$$Y_i(n) = \exp(-\alpha \left\| X(n) - W_i^X \right\|^2 - \beta \left\| Y(n-1) - W_i^Y \right\|^2) \qquad (2.7)$$

The recursive SOM remains close to the original SOM by iteratively applying the algorithm to its own representations. This is an advantage because the SOM is an experimentally proven algorithm that is both successful and biologically plausible.

Therefore, by iteratively applying the original algorithm to its own representations, the recursive SOM is able to process dynamic patterns without having to rely on ad-hoc mechanisms, such as buffers. However, a disadvantage of the recursive SOM is that, like many other TSOMs, it is computationally intensive due to additional connections and neurons. Furthermore, the amount of information fed back via the recurrent connections is directly linked to the number of neurons in the model. Therefore as the size increases, computational efficiency will not increase linearly and will result in large models becoming unfeasible to train.

SARDNET (Sequential Activation Retention and Decay NETwork) (James and Miikkulainen, 1995) is a dynamic extension to the SOM that forms distributed representations of input sequences. The model operates by mapping each input to a best matching neuron, exactly as in the original SOM algorithm. However, once a neuron is designated the winner for a specific input in a sequence, it is precluded from being selected for any other inputs in that sequence (i.e. in the sequence '*BBBB*', each '*B*' will be mapped to a different winning neuron). Additionally, once a neuron is selected as a winner it is assigned an activation value of 1. Upon the presentation of each subsequent input in the current sequence, all activation values are decayed by a predetermined value. This activation and decay mechanism forms a distributed representation of the input sequence on the surface of the map. Experiments on sequences of English phonemes (James and Miikkulainen, 1995) show that SARDNET produces highly descriptive and compact representations.

SARDNET operates in a very elegant and computationally simplistic manner to form distributed representations of input sequences. Furthermore, it has also been shown

that these representations can subsequently be used by supervised neural networks to enhance their performance (Mayberry and Miikkulainen, 1999). However, as discussed in (Hadley and Cardei, 1999), SARDNET does not appear to exhibit strong systematicity (i.e. meaningfully interpreting words in novel positions). Therefore, SARDNET may fail to correctly classify sequences that, while syntactically correct, have a significantly different surface form than those in the training set. This would especially be apparent with sequences involving deep embeddings or recursion of any kind that resulted in sequences longer than those encountered during training.

A notable, but seldom used unsupervised model is ART (Adaptive Resonance Theory) (Grossberg, 1976). This model is a self-organizing pattern classifier that produces classification codes for input patterns. ART typically consists of two layers and a feedback mechanism. The first competitive layer of neurons operates like the SOM and maps an input pattern to a best matching neuron. The output from this layer is used to create an orthogonal vector, with one bit set to denote the winning neuron and all other bits cleared. This orthogonal vector forms the input to the second layer, which produces an activation value that represents ART's classification of the input pattern. During learning, the activation value is fed back and combined (logical AND) with the original input pattern. Weight updates then occur in both layers until the output classification is considered similar enough to the input pattern.

While enhancements have been proposed that enable ART to process more complex types of patterns, in its standard form the model could be applied to similar classification tasks as the SOM. However, an important difference between ART and the SOM relates to the relative organization of internal representations. While the

SOM forms a topological map by clustering similar inputs together, ART does not organize input patterns into any kind of relative structure. Consequently, ART would be unable to capture the type of emergence effects that occur in SOM based models such as context maps (Kohonen, 2001) (i.e. zones of the map representing verbs, nouns etc). Therefore, while ART could be employed as the base for modelling language acquisition, it should be considered only as a second choice to the SOM.

A small number of unsupervised connectionist models are able to solve dynamic problems by extending the principle of Hebbian learning into the temporal domain. Networks such as SOTPAR (Euliano and Principe, 1996) use a temporal Hebbian learning mechanism to correlate the activity between neurons that fire in sequence. While temporal Hebbian learning can be used to extend the SOM into a dynamic model, the mechanism may require full connectivity between neurons, therefore introducing further processing overheads.

2.2.3 Supervised connectionist learning algorithms

The dominant approach to training connectionist models of language acquisition involves supervised training algorithms. While the roots of this approach can be traced back to the work of McCulloch and Pitts (1943), the majority of models have only emerged following the key publication (Rumelhart *et al.* 1986). Rumelhart and Hinton's error back-propagation learning algorithm showed how connectionist models could be trained to solve non-linear problems. The fundamental idea behind supervised learning is that the error signal (i.e. the difference between the model's response to an input and the desired response) is used to modify the weights in order

to reduce the future error. Thus the model's individual weights are effectively punished and rewarded until they reach the correct values to represent the problem.

The advantage of supervised learning is that a model can be trained using only a subset of inputs and desired output pairs for a particular problem. From this subset of training data, the learning algorithm may be able to construct an approximate solution to the problem and subsequently generalize to unseen inputs. Furthermore, unlike their unsupervised counterparts, most supervised connectionist models use distributed representations. By using multiple weights to represent each input pattern, a distributed model does not rely on any single weight to hold a specific piece of information. Consequently models using distributed representations are efficient and fault tolerant. They also exhibit graceful degradation when connections are damaged or removed. Supervised connectionist models that use distributed representations are commonly referred to as Parallel Distributed Processing (PDP) models.

2.2.3.1 Supervised connectionist models of language acquisition

Since the connectionist renaissance starting in the late 1980s, the majority of the NLP models published in the literature have been applied to language acquisition. The first notable such model was published by Rumelhart and McClelland (1986) and involved the acquisition of the markings of the English past tense. Accounting for the process by which children learn the English past tense is a common battleground for competing linguistic theories. The Rumelhart and McClelland (1986) model maps a representation of the present tense of an English verb onto the equivalent representation of that verb's past tense using a multi-layer perceptron (MLP). Its authors claim this model is capable of mimicking the U-shaped learning curve

characteristic of child language learners (fig 2.5), but without the need for explicit rules. However, the model has been heavily criticized in a number of areas (Fodor and Pylyshyn, 1988; Pinker and Prince, 1988), including its input representation, its erroneous predictions of novel morphological derivations and its artificial training regime. A number of subsequent researchers (Plunkett and Marchman, 1996; Jackson *et al* 1996) have attempted to overcome the limitations of the original Rumelhart and McClelland (1986) model by, for example, employing more realistic training regimes.

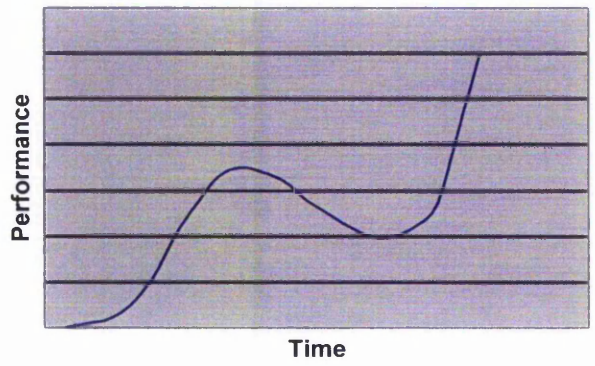


Figure 2.5 – U-shaped learning curve of child performance

One of the most influential publications in the field of connectionist language acquisition since the initial Rumelhart back-propagation algorithm was that of Elman's SRN (Elman, 1990). The Simple Recurrent Network (SRN) is a dynamic extension of the MLP which uses recurrent connections to feedback the hidden layer activations at the next time step. In a similar manner to their unsupervised counterparts (section 2.2.2.2), this recurrency mechanism allows the SRN to process sequences of inputs. In (Elman, 1990) the SRN was applied to the acquisition of syntactic structure. These experiments involved training the model to predict the next word in a sentence that was randomly generated from a simple grammar. While the model was never able to reliably predict the exact next input, due to the non-

deterministic nature of the task, its attempts to do so allowed it to induce a representation of the underlying structure of the grammar. Thus after training the model would be able to predict the category of the next word in a sentence (i.e. plural-noun or singular-verb). The SRN has subsequently been used in further grammar induction problems (Cleeremans *et al,* 1989; Sharkey *et al,* (2000) which have further investigated its ability to learn the structure of simple grammars. However, the degree of grammatical knowledge learnt by the SRN is highly controversial. Mainstream linguists such as Jackendoff (2002) and Marcus (2000) argue that the model has only learnt surface regularities and is incapable of learning a meaningful representation of a grammar.

An important area of language acquisition involves the initial process of learning individual words. The complexity of this process of segmenting continuous speech into individual words can be perceived by listening to a native speaker of a foreign language. Unlike a person's native language, sentences of speech in a foreign language are initially perceived as a continuous stream, rather than a collection of discrete words. However, children learn to solve this segmentation problem and have been shown to acquire specific phonetic characteristics of their native language by the time they are six months old (Davis, 2003).

The problem of speech segmentation has been modelled using a number of connectionist simulations. Aslin *et al* (1996) presented a supervised connectionist model that used phoneme trigrams to identify the boundaries between utterances in a corpus of child-directed speech. Elman (1990) applied the SRN to a segment prediction task in which a continuous stream of phonemes, corresponding to words in

an artificial grammar, was presented to the model. Variations in the SRN's prediction error suggested that the model had learnt the combinations of phonemes that constituted words and could thus identify word boundaries via its inability to accurately predict the phoneme following a word. However, larger scale experiments (Cairns *et al,* 1997) have called into question Elman's predictive phonetic word segmentation approach by showing that only 21% of actual word boundaries can be predicted.

A problem closely associated with speech segmentation is that of vocabulary acquisition. Once children are able to recognise the boundaries separating words, how do they learn to pair these words with their associated meaning? Vocabulary acquisition involves the process of mapping sequences of speech phonemes onto an associated lexical or semantic representation (Davis, 2003). Initial connectionist models of vocabulary acquisition performed one-to-one mappings between specific words in the input stream and an associated target representation (Plunkett *et al,* 1992). However, such an approach assumes that there is sufficient information in the child's learning environment to perform a one-to-one mapping between specific words and their meaning. In order to make the task more realistic, subsequent connectionist experiments performed mappings between entire sequences and associated meanings (Davis, 2003).

An important model with regard to both connectionist representational schemes and natural language is Pollack's RAAM (Recursive Auto-Associative Memory) (Pollack, 1990). This model uses a recursive auto-associative memory that allows the encoding and recall of variable sized sequences or tree structures. This model answers many of

the critics of connectionism (Minsky and Pappert, 1969; Fodor and Pylyshyn, 1988) who argued that the input representations used in connectionist models were insufficient to model cognitive structure. RAAM has been applied to language acquisition in a number of experiments. In the original paper (Pollack, 1990), the RAAM model was applied to learning propositional sentences. It was shown that RAAM produced internal representations that allowed the model to recognise and process novel sentences. Many other researchers have subsequently used RAAM to model aspects of the syntactic structure of natural language (Chalmers, 1990; Blank *et al,* (1991).

2.3 Limitations of connectionism

Connectionist models have been applied to a variety of aspects of language acquisition, from inflectional morphology (Rumelhart and McClelland, 1986), to grammar induction (Elman, 1990). However, many traditional linguists have criticized the results of these experiments and questioned the applicability of connectionism to language acquisition (Fodor and Pylyshyn, 1988; Jackendoff, 2002).These arguments have centered on issues of biological plausibility, adaptive generalization, scalability and psychological similarity to human learners.

2.3.1 Arguments against biological plausibility

The most common arguments against connectionism involve the perceived decreasing relation between artificial neural networks and their biological counterparts. These arguments stipulate the field's loss of focus from its founding tenet of biological plausibility. Connectionist models such as the MLP (Rumelhart *et al,* 1986) aren't

realistic models of the structure, the individual neurons or even the learning process that occurs in biological neural networks (Sejnowski, 1986).

While aimed at connectionism in general, these arguments are, in most cases, specific to the commonly used back-propagation learning algorithm and PDP models. While the biologically plausibility of back-propagation has polarized the connectionist community itself, the algorithm is without a doubt the single most important advancement in connectionist modeling. When Rumelhart *et al* (1986) introduced back-propagation it provided a solution to non-linearity, a problem that had relegated early connectionist models to obscurity (Minsky and Papert, 1969). This allowed multi-layer models to be created to learn linearly-inseparable problems, from the XOR logic function to the acquisition of the English past tense.

The argument against back-propagation involves the error return signal which must flow backwards through the network to every non-output layer neuron. This means that the algorithm uses non-local information to update the weights for each neuron, an operation that is at odds with neuro-biological research[3] (Hebb, 1949; Kohonen, 2001). Further arguments involve the sensitivity of multi-layer back-propagation networks to the number of hidden layer neurons, which must be known a-prior in order to generalize properly (McLeod *et al*, 1998). While most connectionists accept these arguments, they defend the use of back-propagation by claiming that it encodes representations in the weights in a similar distributed manner to that of biological neural networks. Thus, the algorithm may not operate like its biological counterpart, but it does produce models that have a claim to biologically plausibility.

[3] See p. 116-117 in (McLeod *et al*, 1998) for a more detailed explanation of the plausibility issue.

2.3.2 Arguments against connectionism for developmental cognitive modelling

One of the most fundamental and understated problems for connectionist models of cognition is that of variables. Conventional feed-forward connectionist models represent activity as a spread of activations values through the network. Such models are incapable of encoding the type of variables that could represent relationships such as X equals Y (Marcus, 1998). Typed variables that allow the abstract treatment of classes of a particular word or linguistic rule are fundamental to the combinatorial nature of language (Jackendoff, 2002). Without the power to manipulate an abstract representation of *everything of type X*, relation problems such as learning 'which X rhymes with which Y' are unsolvable.

Due to the relative lack of research into unsupervised connectionism, it is unclear whether such models are capable of either learning or representing variables. However, most unsupervised models use localist architectures and therefore possess discrete internal states that could be used for the representation of variables. Consequently, assuming that a suitable learning algorithm was employed, such models may be capable of solving problems that require the use of variables.

Another understated problem regarding models trained using gradient-descent learning algorithms is their sensitivity to initial starting states. Experiments by Kolen and Pollak (1990) found that the effect of random initial starting weights had a dramatic effect on the model's ability to converge on a solution. Further experiments on SRNs by Sharkey *et al* (2000) also found that the models were extremely sensitive to their initial starting weights and that only one in every forty-five models was

actually able to solve the given problem at all. This poses a problem for a connectionist account of language acquisition because, with few exceptions, all children are able to learn their native language.

A central argument against connectionist models relates to their adaptive generalization abilities. As explained by Sharkey *et al* (2000), if connectionist models are to be used to model human cognition they must exhibit similar developmental properties to those observed in humans. Grammatical-transfer experiments (Sharkey *et al,* 2000) show that the SRN is unable to exploit previous grammatical knowledge. These experiments show that when a model trained on a specific grammar is exposed to new lexical items, training times are adversely affected. Such findings are at odds with human performance, for which intuition suggests that language acquisition should get easier as development progresses.

The ability to perform grammatical-transfer is inter-related to another undesirable behavior that occurs in gradient-descent based connectionist models. The inability to retain knowledge across training sets, known as *catastrophic forgetting* (French, 1999), entails that learnt-knowledge must be continually refreshed by cycling through the entire data set. Such behavior is psychologically implausible because children are able to learn new knowledge without necessarily overwriting existing knowledge (Sharkey *et al,* 2000).

Another important argument leveled against connectionist models of cognition concerns their ability to learn and generalize from sparse data. Fodor and Pylyshyn's original criticism (1988) concerning connectionist generalization are formalized by

Hadley's definition of strong systematicity (Hadley, 1994). The premise of this argument is that children are capable of learning language without encountering words in all of their syntactically legal positions. Hadley argues that connectionist models, such as the SRN and RAAM, use positionally intensive training regimes in which words are encountered in all syntactically legal positions (Hadley and Cardei, 1999). Therefore it is argued that such models are not cognitively realistic because they don't learn from the type of sparse data that children use to acquire language.

While there are a number of serious limitations concerning supervised connectionist models of language acquisition, there are also a number of fundamental limitations constraining unsupervised approaches. The primary limitation of unsupervised connectionism is the lack of effective models and training algorithms, especially those capable of tackling the type of dynamic sequences found in language. The majority of unsupervised connectionist models that are applicable to language acquisition are highly computationally intensive, in certain cases requiring supercomputer resources (Voegtlin and Dominey, 2002). In most cases this complexity is a result of localist architectures which impose prohibitive memory and processing constraints on large scale models.

2.3.3 Learning deterministic representations using a continuous state-space

Since the early nineties connectionist models, specifically the SRN, have been applied to the problem of grammar induction. These experiments, which have involved simple regular and context-free grammars, have met with some success (Elman, 1990; Cleermans *et al*, 1989), suggesting that supervised connectionist models can learn to emulate finite-state automata. However, detailed analysis of models trained on these

tasks show a number of fundamental problems that derive from using a model with a continuous state-space to approximate a discrete problem.

While supervised connectionist models are capable of learning simple formal languages, they are renowned for their instability when processing long sequences that were not part of their training set (Kolen, 1994; Omlin, 2001). A model such as the SRN is capable of partitioning its state space into regions that are believed to approximate the states in a grammar. However, sensitivity to initial conditions means that each transition between regions of state space will result in a slightly different trajectory (Kolen, 1994). This causes instability when transversing state trajectories that were not seen during training. Such instabilities arise due to slight discrepancies in the trajectories that are compounded with each transition until they exceed the locus of the original attractor, resulting in a transition to an erroneous region of state space.

Such behavior is characteristic of supervised dynamic connectionist models and can be seen as both a power and a weakness of this class of models. While this representational power enables the model to surpass deterministic finite automata and emulate non-deterministic systems, it proves to be a significant disadvantage when attempting to emulate the deterministic behavior fundamental to deterministic finite automata (DFA). Attempts have been made to produce discrete state-space models by using a step-function for the hidden layer neurons (Zeng *et al*, 1993). However, while this technique eliminates the instability problem, the use of a non-differentiable function means that the weight-update algorithm's sigmoid function can only approximate the error signal. This weakens the power of the learning algorithm,

increasing training times and in some cases causing the model to learn an incorrect representation of the DFA (Omlin, 2001).

Other notable techniques for overcoming instability in continuous state-space models include the Simple Synchrony Network (SSN) (Lane and Henderson, 1998), which utilizes Temporal Synchrony Variable Binding (TSVB) to encode entities using pulsing binary-threshold neurons. This technique is able to enhance the power of continuous state-space models by providing static building blocks within the ever changing sea of internal representations. Given the level of research that has gone into connectionist variable binding (Browne and Sun, 2000) and stability issues (Omlin, 2001), a potentially desirable characteristic of unsupervised connectionist models is that their localist architectures provide the potential for a discrete state-space.

2.4 Discussion and conclusions

Language is a complex and powerful system that describes and perhaps even shapes every aspect of human perception (Sapir, 1929; Gordon, 2004). However, the process of language acquisition itself is a paradox. While children appear to engage in a process of learning their native tongue, the linguistic input they are exposed to appears too sparse to permit acquisition of a grammar (Chomsky, 1965). Traditional theories of linguistics have therefore assumed a certain level of innate knowledge that constrains language acquisition and equips children with a prior knowledge of grammatical structure. However, despite the apparent intractability of the problem, the allure of an automated language acquisition system has fuelled research for half a century.

Recent advances in connectionism have challenged the views of traditional linguists by proving that linguistic input has a far richer structure than was previously believed. The popular connectionist supervised learning paradigm has produced small scale models addressing all areas of language acquisition, from lexical segmentation (Aslin *et al,* 1996) to grammar acquisition (Cleeremans, 1989; Elman, 1990). However, a number of potentially fundamental problems have been identified with these models. The inability to represent variables prevents these models from representing abstract relationships, such as those found throughout language (Marcus, 1998). Furthermore, their stability, scalability and biological plausibility have been used to question the applicability of connectionism as a tool for modelling language acquisition (Sharkey *et al,* 2000).

While research has focused on the popular paradigm of supervised connectionist learning, much less attention has been given to unsupervised models of language acquisition. Unsupervised learning involves simple, local operations that are directly inspired by neuro-biological evidence from the human brain and nervous system (Hebb, 1949; Kohonen, 2001). Unsupervised models, such as the SOM emulate the operation of topological maps shown to exist in multiple areas of the cortex (Kohonen, 2001). However, despite their biological plausibility and data-orientated modelling approach, relatively few unsupervised connectionist models of language acquisition have been proposed. The few notable models which have been developed (MacWhinney, 2002b) suggest that unsupervised learning may provide the stability by which connectionist language acquisition can scale up to larger problems. Such large scale models may potentially involve hybrid approaches that combine the

stability and simplicity of unsupervised learning with the more general purpose capabilities of supervised learning algorithms.

One of the reasons for the limited application of unsupervised connectionist models in language acquisition may be related to the static nature of popular models such as the SOM. While the successful supervised connectionist models have dynamic capabilities, relatively few viable unsupervised dynamic models have been proposed. A chronological comparison shows that unsupervised dynamic models have so far followed similar developmental phases as their supervised counterparts. While early unsupervised dynamic models used buffers to map time onto space, later models represented time by the effect it had on processing. Given this relationship, an obvious area of future research into unsupervised modelling would involve recurrency mechanisms. So far only one significant unsupervised model uses recurrent connections (Voegtlin and Dominey, 2002) and that model is limited by the size-computational complexity issue.

In conclusion, analysis of unsupervised connectionist modelling has shown that their discrete state-spaces and simple mode of operation provide the potential for modelling the finite-state properties of language in a more robust and biologically-plausible manner than existing supervised models. The discrete state-space properties of localist architectures, such as the SOM, provide similar representational power to that of symbolic algorithms. This has already provided such unsupervised models with the stability to scale up to problems beyond the capability of their distributed counterparts (MacWhinney, 2002b). Such properties may also potentially allow unsupervised

models to overcome the generalization instabilities that plague current supervised models (Kolen, 1994; Omlin, 2001).

3 – Simple Recurrent SOM (SRSOM)

3.1 – Introduction

The literature study has revealed that language acquisition is a highly complex process which is interpreted by many traditional linguists as a paradox. Such nativist views have been rejected by empiricists, who have sought to explain language acquisition using the recent cognitive modelling technique known as connectionism. However, the use of these biologically inspired learning models has not proved to be the panacea that many researchers had hoped for. A number of serious limitations involving issues such as the representation of variables, adaptive generalization abilities and plausibility have constrained connectionist models from scaling up to large scale language acquisition (Marcus, 1998; Jackendoff, 2002). While these limitations are based upon the capabilities of supervised models, the alternative class of unsupervised connectionist learning algorithms may offer the means to create larger-scale, more biologically plausible models of language acquisition.

The review of notable unsupervised models in the literature study showed that a few models existed which incorporated the dynamic memory mechanisms required to process the type of temporal sequences found in language. However, analysis showed that all of these models possessed various deficiencies that rendered them inappropriate for modelling language acquisition. The main limitation of the more powerful models such as the Hypermap and the Recursive SOM is the relationship between their computational intensity and their size. As these models increase in size, the resources required for memory and processing increase non-linearly. Consequently, the direct use of these existing models was deemed inappropriate for

an investigation into language acquisition. In order to determine whether unsupervised connectionist models could capture the finite-state properties of language, it was determined that a new model must be developed. This model must be capable of processing temporal sequences in an unsupervised manner, but with a computational efficiently that wasn't prohibitive to large scale modelling of language acquisition.

3.2 – Proposed new model

The SOM was chosen as the base model for this research because the majority of unsupervised dynamic connectionist models use it to extend their processing capabilities into the temporal domain. As described in section 2.2.2, the SOM uses a winner take all learning paradigm to map a distributed input vector onto a best matching neuron in a, usually rectangular, grid of neurons. However, the SOM is designed to process spatial data, such that the choice of winning neuron is not directly affected by any preceding data processed by the network. Therefore in order to allow the proposed model to process temporal data a recurrency mechanism must be employed. As with the Recursive SOM, this recurrency mechanism will allow the choice of winning neuron to be based not just on the current input, but also on some representation of past context.

In order to provide the model with a computationally minimized representation of context, the proposed recurrency mechanism feeds back a representation of the relative map location of the previous winning neuron (fig 3.1). Given that the winning neuron is a best matching representation of the input vector, the inclusion of the previous winner's location in the selection algorithm allows the concept of *best*

matching neuron to be applied recursively to an entire sequence. The winning neuron is therefore potentially a unique representation of the current input and the entire preceding sequence of inputs. The temporal depth and resolution of such a recurrent feedback mechanism is potentially infinite, being limited only by the size of the map. Furthermore, compared to other dynamic SOMs it is a very efficient method of representation because each winning neuron can be represented using two numbers. The new model presented here will be referred to as the SRSOM (Simple Recurrent SOM).

3.2.1 - Architecture and algorithm for the SRSOM

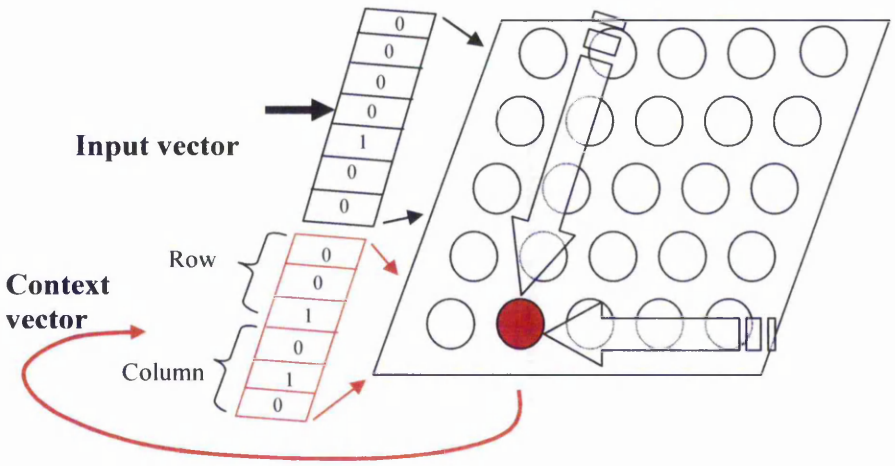


Figure 3.1 – Diagram of SRSOM showing feedback of the previous winning neuron's column and row.

The recurrency mechanism for the SRSOM operates by feeding back two binary numbers which represent the column and row of the previous winning neuron. The size of the context vector is determined dynamically based on the number of neurons in the model (i.e. the context vector is large enough to represent the locations of all neurons in the map). This provided an optimal representation of context as it employs

the minimum possible number of bits in order to uniquely represent every possible neuron. For example, for an SRSOM with 25 neurons, such as that shown in fig 3.1, the context vector would need to consist of six bits. This representation then provides a unique context representation for all 25 neurons in the model. The context vector is treated as an implicit part of the input vector, allowing the model's learning algorithms (equ. 3.1-3.4) to remain identical to those of the standard spatial SOM. These algorithms consist of selection of the winning neuron (equ. 3.1), weight update (equ. 3.2), neighbourhood function (equ. 3.3) and a Pythagorus-derived distance calculation (equ. 3.4). This latter function is used as part of the neighbourhood function (equ. 3.3).

- $N = \arg(\min_j(\sum_i |X_i - W_{ij}|))$

Equation 3.1 – Winner selection algorithm

- $W_{ij}(t+1) = W_{ij}(t) + \alpha h_{ij}(X(t) - W_{ij}(t))$

Equation 3.2 – Weight update algorithm

- $h_{ij} = \exp(\frac{-d_{ij}^2}{2\sigma^2})$

Equation 3.3 – Neighbourhood function

- $d_{ij} = sqrt((Column_i - Column_j)^2 + (Row_i - Row_j)^2)$

Equation 3.4 – Algorithm to calculate the distance between two neurons

As explained in section 2.2.2, the winning neuron N is defined as the neuron whose weight vector W_{ij} is the best match for the current input X_i (equ. 3.1). Once the winning neuron has been selected its weights, along with those of neighbouring neurons, are updated $W_{ij}(t+1)$ (equ. 3.2) to make them a better match for the current

input X(t). This weight update operation involves calculating the new weight vector $W_{ij}(t+1)$ using the original weight vector $W_{ij}(t)$, the learning rate α and the value of the neighbourhood function for the neuron in question. The learning rate α determines the level of weight change. It should initially be set to around 0.1 and decreased linearly each epoch throughout training (Kohonen, 2001).

The neighbourhood function (equ. 3.3) uses the Pythagorean distance between the neuron in question and the winning neuron in order to make the level of the weight update operation proportional to the distance from the winning neuron. The symbol σ in equation 3.3 controls the width of the neighbourhood function and determines how many neurons are affected by the neighbourhood function. σ should initially be set to half the size of the map and it should be decreased linearly every epoch throughout training (Kohonen, 2001) (i.e. in a 10×10 model σ should initially be set to 5). In equation 3.4 the variables $Column_i$ and Row_i denote the column and row of the winning neuron (ex. 2 and 1 respectively in fig 3.1), while $Column_j$ and Row_j denote the column and row of the neuron that the neighbourhood function is being applied to.

3.2.2 – Initial experiments to test recurrency mechanism

Once the new feedback mechanism had been implemented, a clustering analysis was performed using simple sequences consisting of four 4-bit elements (table 3.1). The first element of each of these five simple sequences is unique, while the following three elements are identical. The sequences were selected in order to explore the effect of varying context upon the choice of winning neuron (i.e. how will different initial winning neurons affect the choice of subsequent winning neurons?). This

analysis involved training a 10 × 10 SRSOM over 1000 epochs using an initial learning rate of 0.1 and initial neighbourhood of 5.

The intended purpose of the SRSOM is to cluster similar input patterns together on the map. While the original SOM clustered spatially similar patterns together, it is intended that the SRSOM should organize itself based both on the input's spatial and contextual representations. Therefore the expected outcome of this clustering analysis was that the second elements in sequences that had similar contexts, due to the proximity of the first winning neurons in the sequence, would be clustered near each other on the map.

Table 3.1 shows the results from the clustering analysis for the five simple sequences. These results show that the SRSOM has clustered the second elements from sequences one and two to the same neuron. The same occurs for the second elements from sequences three and five, which both share another neuron. In both these cases where the second elements clustered to the same neuron, the first elements from the respective sequences were located close to each other on the map.

These results were analysed by looking at the contextual representations for the first winning neurons in the sequences. The far right column of table 3.2 shows the binary contextual representations for the first winning neurons in the sequence. The most significant five bits (left) of these values consist of a binary number representing the neuron's column, while the least significant five bits (right) represent the neuron's row.

	Element 1 (Winning neuron)	**Element 2** (Winning neuron)	Element 3 (Winning neuron)	Element 4 (Winning neuron)
Sequence 1	**1111 (11)**	**0010 (90)**	0100 (81)	1000 (40)
Sequence 2	**0111 (1)**	**0010 (90)**	0100 (81)	1000 (40)
Sequence 3	**0001 (5)**	**0010 (98)**	0100 (93)	1000 (20)
Sequence 4	**0010 (58)**	**0010 (78)**	0100 (93)	1000 (20)
Sequence 5	**0100 (25)**	**0010 (98)**	0100 (93)	1000 (20)

Table 3.1 – The elements and corresponding winning neurons for five sequences which were clustered using the SRSOM operating using a binary context representation.

The representations for neurons 11 and 1 (sequences 1 and 2 in table 3.1) both have the same bit set in the latter five bits of their representation. This is because both neurons 11 and 1 share the same row and therefore have the rightmost bit set to indicate row one. This is also the case for neurons 5 and 25 (sequences 3 and 5 respectively) which share the same latter five bit representation. This sharing of neurons suggests that the SRSOM is able to exploit the context representation and cluster inputs based on similarity in context, as well similarity in form. This is exactly what the SRSOM was intended to do because such contextual clustering may allow it to discover the structural information underlying language.

	Element 1	Winning neuron	2D binary representation of winning neuron
Sequence 1	**1111**	**11**	**00010 00001**
Sequence 2	**0111**	**1**	**00001 00001**
Sequence 3	**0001**	**5**	**00001 00101**
Sequence 4	**0010**	**58**	**00110 01000**
Sequence 5	**0100**	**25**	**00011 00101**

Table 3.2 – Representation of the winning neurons for the 1st elements of each sequence. Shown using a binary context vector representation.

3.2.3 Enhanced Graycode context representation

Following further analysis of the results from the initial clustering experiment, a potential theoretical design flaw was discovered. As previously discussed in section 3.2.2, the recurrency mechanism used two binary numbers to represent the column and row of the previous winning neuron. This representation was intended to ensure that changes in the Hamming distance (the number of bits that are different) between any two neurons' coordinate representations was proportional to the linear distance between those two neurons on the map. However, it was found that due to the binary representation used in the context vector, the Hamming distance between two neurons' context representations didn't increase smoothly as the distance between those two neurons increased. Instead the Hamming distance for certain relatively distant neurons (e.g. 1 and 7) is lower than for neighbouring neurons (table 3.3).

Neuron number	Binary context vector representation	Hamming distance
7	**00001 00111**	
1	**00001 00001**	2
7	**00001 00111**	
8	**00001 01000**	4

Table 3.3 – Hamming distance for two sets of neurons. Due to the binary representation scheme the hamming distance is greater for adjacent neurons 7 and 8 than for neurons 1 and 7, which are located physically further apart on the map.

Such a representational weakness could seriously affect the models' ability to cluster temporal patterns. The binary representation used in the coordinate feedback system was therefore changed to a Graycode (Gray, 1953) representation. Unlike the binary number system, the Hamming distance between any two consecutive Graycode numbers is equal to one. Thus by using a Graycode representation for both coordinate vectors, the Hamming distance between any two neuron's 2D coordinate representations will be more representative of the physical distance between those two neurons on the map.

The revised Graycode context representation was tested using an identical clustering experiment as that employed in the previous section. As shown in table 3.4, these results show that as with the previous experiment, inputs with similar contexts are being clustered together to similar locations on the map, exactly as expected. However, the results in table 3.4 also show that the third elements in sequences one and two are also being clustered to similar neurons. This shows that the revised Graycode context representation has allowed the model to identify similarities between neurons of close physical proximity.

	Element 1 **(Winning Neuron)**	**Element 2** **(Winning Neuron)**	Element 3 (Winning neuron)	Element 4 (Winning neuron)
Sequence 1	**1111 (92)**	**0010 (21)**	0100 (17)	1000 (93)
Sequence 2	**0111 (91)**	**0010 (12)**	0100 (18)	1000 (94)
Sequence 3	**0001 (40)**	**0010 (44)**	0100 (10)	1000 (72)
Sequence 4	**0010 (97)**	**0010 (57)**	0100 (15)	1000 (78)
Sequence 5	**0100 (50)**	**0010 (41)**	0100 (28)	1000 (85)

Table 3.4 – Elements and corresponding winning neurons for five sequences clustered using the SRSOM operating with a 2D Graycode context representation.

As can be seen from table 3.5 the binary representations for the second winning neuron (i.e. the context for those third elements) are very different, despite the fact that the neurons are adjacent. The difference between the representations for these neurons further highlights the weakness in the binary representation scheme. However, as can also be seen from table 3.5 the corresponding Graycode representation solves this problem because Graycode values differ in only two bits.

Neuron	Binary representation	Graycode representation
21	00111 00001	0010 0001
12	00000 00010	0000 0011

Table 3.5 – Binary and Graycode representations for two winning neurons.

These results show that the Graycode context representation provides an improvement over the equivalent binary representation scheme. The use of Graycode improves the

network's ability to cluster inputs whose contexts would be misrepresented using a binary scheme.

3.3 - Experimental analysis of SRSOM on simple grammars

3.3.1 – Connectionist grammar induction

In light of the success of the initial clustering analysis it was decided that the SRSOM should be applied to a real linguistic problem. Grammar induction is the machine learning problem of modelling an unknown grammar using a finite set of positive (and possibly also negative) examples generated by the grammar. In computational linguistics, the usual measure of a model's ability to learn a grammar is via its competence at predicting the next possible symbols following each input in a sentence (Cleeremans *et al*, 1989; Elman, 1990). In order to compare the SRSOM model against the SRN (a popular supervised model), it was decided that the SRSOM should be given the task of predicting the next possible symbols in the grammar. Such a prediction task has been used in multiple grammar induction investigations involving supervised connectionist models (Cleeremans *et al*, 1989; Elman, 1990).

The grammar chosen for this experiment was the Reber grammar. This simple regular grammar was originally devised to investigate implicit rule learning in human subjects. It has since been used by various researchers (Cleeremans *et al*, 1989; Sharkey *et al* 2000) for investigating connectionist grammar induction using the SRN. The Reber grammar is a member of a class known as regular grammars. As discussed in detail in section 2.1.1, regular grammars occupy the lowest level of the Chomsky hierarchy. They are characterized by their simple production rules in which the left hand side consists of a single non-terminal while the right hand side of the production

rule can have no more than one terminal symbol. While regular grammars are a gross simplification of natural language, they do contain important linguistic characteristics, such as the use of generative production rules and recursion. Therefore, due to their simplicity regular grammars are a good *test bed* for the development of linguistic models.

The advantage of applying the SRSOM to the same Reber grammar problem, as used to evaluate the SRN (Cleeremans *et al*, 1989), is that a direct comparison can be made between the performance of the two models. If it can be shown that the SRSOM is able to learn the Reber grammar to a comparable extent as the SRN, then this will show that unsupervised connectionist models of language acquisition perform on a comparable level to their supervised counterparts.

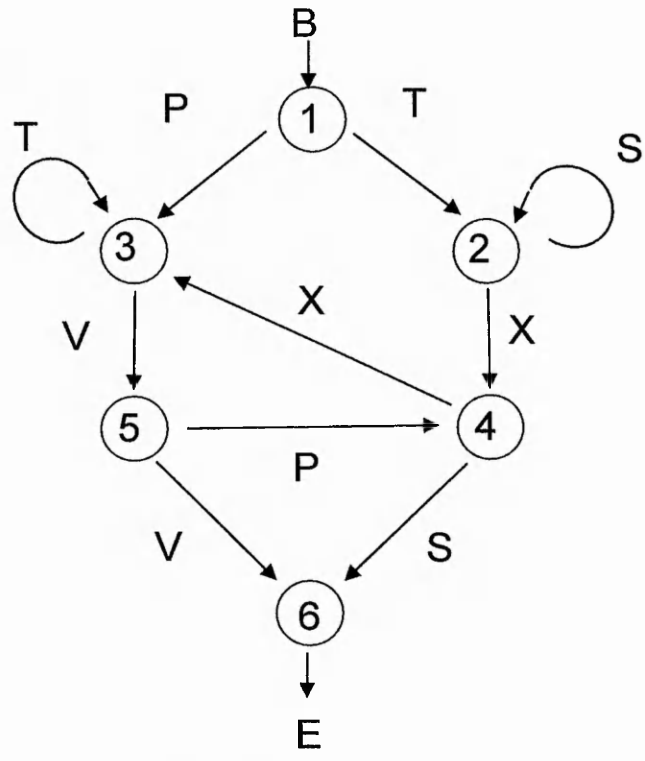


Fig 3.2 – Finite-state machine for the Reber grammar

Fig 3.2 shows the finite-state machine (FSM) for the Reber grammar. The numbered circles show states in the grammar, while the arrows and respective letters represent the state transitions. Sentences are generated by starting from state 1 and moving from one state to another until state 6 is reached. Table 3.6 shows example sentences generated by this grammar. States two and three have recursive loops which allow the FSM to move back into the same state. This theoretically allows the generation of an infinite number of unique sentences. In the experiments performed here, the maximum number of times the FSM could go around a recursive loop was limited by the *recursive depth* parameter. The purpose of this parameter was to limit the sequences generated by the FSM so that the training sequences would not be of a far greater recursive depth than the test sequences.

B->T->S->S->X->S->E
B->P->V->P->X->T->T->V->V->E
B->T->S->S->S->X->X->T->T->V->P->X->T->V->P->S->E

Table 3.6 – Example sequences generated by the Reber grammar.

3.3.2 – An SRSOM grammar inductor

In order to induce the Reber grammar, the SRSOM must be able to both process input symbols in a sequence and also test its induced knowledge. The first requirement was met by adopting an orthogonal input vector representation. This common representation scheme is used throughout connectionist linguistics (Elman, 1990) and involves representing each symbol in the grammar by setting the appropriate bit in the vector to one and setting all the other bits to zero (table 3.7). The advantage of using

orthogonal input vectors is that they prevent the model from discovering any potentially misleading form-based similarities between input symbols (e.g. *widow* and *window* are similar looking words, but have unrelated meanings). Orthogonal input representations are also quite biologically plausible. It has been shown that competitive learning, which is believed to operate throughout the brain and nervous system, can produce output patterns that are less correlated than the corresponding input patterns. This is achieved by mapping similar input patterns to multiple output neurons (McLeod *et al*, 1998).

Grammatical symbol	Orthogonal input vector representation
B	0 0 0 0 0 0 1
T	0 0 0 0 0 1 0
P	0 0 0 0 1 0 0
S	0 0 0 1 0 0 0
X	0 0 1 0 0 0 0
V	0 1 0 0 0 0 0
E	1 0 0 0 0 0 0

Table 3.7 – Orthogonal input vector representation for SRSOM.

A common approach to testing the performance of models in connectionist grammar induction involves predicting the next valid symbols in the grammar (Cleeremans *et al*, 1989; Elman, 1990). During the operation of processing symbols in a sequence, the SRSOM uses the location of the previous winning neuron as context. Therefore, it is possible to predict the next winning neuron in a sequence by finding the best matching neuron whose context vector (NOT symbol vector) matches the location of

the current winning neuron. Fig 3.3 shows a simple example of this prediction operation.

By using the proposed prediction algorithm, the second winning neuron can be predicted by taking the first winning neuron's location (column 1, row 10) and finding the neuron whose context vector best matches this value (i.e. has the lowest Euclidean distance). In fig 3.3, the neuron in column 10, row 2 (the 2nd winning neuron) has the context vector that best matches the location of the first winning neuron. Therefore, the symbol vector for the neuron at column 10, row 2 would become the predicted next symbol (i.e. { 0, 0, 0, 0, 0, 1, 0 }).

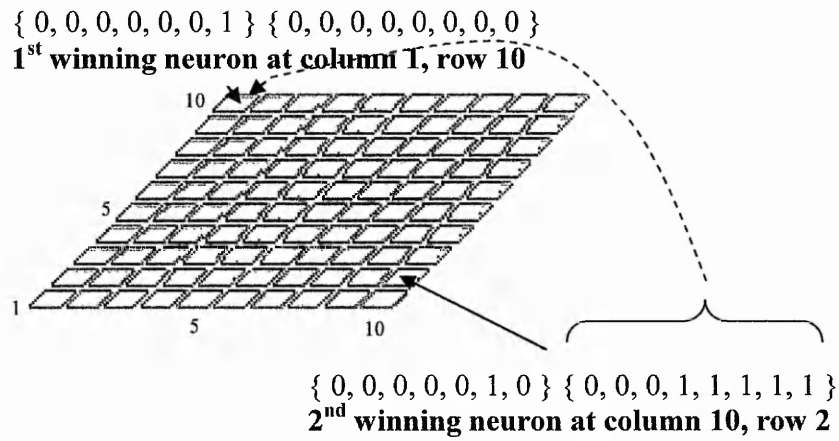


Fig 3.3 - Example showing operation of the prediction algorithm. The neuron at column 10, row 2 can be predicted as the next winner because its context vector contains the location (shown via the dotted line) of the 1st winning neuron 1,10. Therefore, the symbol vector for neuron at column 10, row 2 becomes the predicted next symbol (i.e. { 0, 0, 0, 0, 0, 1, 0}).

This method of prediction is repeated to find the neuron with the second best matching context vector in the map. However, in order for this second best matching neuron to be selected as the second predicted next symbol, it must represent a different symbol than the first predicted next winner. For example, following an initial 'B' input, if the model's first prediction is a 'T' symbol (as in fig 3.3) then its second

prediction must be a 'P' symbol in order for the prediction to be considered correct. Therefore, by repeating the prediction process and applying the simple criterion of requiring non-duplicate symbols, the SRSOM is able to generate two predicted next symbols. These predicted symbols can then be tested against the next valid symbols in the grammar to assess the model's performance.

In order to quantify the SRSOM's ability, the accuracy of the model is expressed as a percentage of the number of predictions it got right out of the total number of predictions made. As with the SRN experiments (Cleeremans *et al*, 1989), the SRSOM is tested on a number of sequences that are generated on the fly. For each sentence a symbol is presented and the SRSOM must then predict the next possible symbol in the sequence. At each step the model makes two predictions about what the next symbol will be. The reason for choosing two predictions is that at every state in the grammar there are at least two possible next valid symbols. If both of these predictions are valid next symbols in the grammar, then the prediction is considered correct. However, if either of the predictions is not a valid next symbol in the grammar, then the prediction is considered incorrect.

3.3.3 Experiments on the Reber grammar

This section of the report will present results from a series of experiments conducted on the SRSOM operating with the new Graycode context vector. The performance of the model was assessed based on its accuracy at learning the Reber grammar, as discussed in the previous section. The purpose of these experiments was two-fold. First, the experiments would show to what degree the SRSOM was able to learn the grammar and what level of embedding (number of sequential recursive symbols) it

was capable of processing. Additionally, the experiments would investigate what resources (i.e. model size, number of epochs etc) were required and whether increasing these resources would significantly affect performance.

3.3.3.1 Generation of training and test sets

In the grammar induction experiments performed by Cleeremans *et al* (1989), the SRN was trained on sequences randomly generated *on the fly*. However, there is a widely held consensus among connectionist researchers that training and test data should be selected in a more scientifically rigorous manner than just *on the fly* generation (Hopgood, 2001). Such researchers advocate the use of strictly separate training and test sets to ensure a model is tested on data it hasn't encountered during training. By using previously unseen data for testing, the model's performance is a more accurate measure of its ability to generalize. Furthermore, by creating separate fixed training and test sets in advance, detailed analysis of the training process becomes possible (i.e. does the use of short sequences improve performance?). Consequently, it was decided that the SRSOM should be assessed using randomly generated separate training and test sets.

While separate training and test sets are more rigorous than an on-the-fly approach, they do compound what is known as the *sparse data problem* (Chomsky, 1965). This may occur in an infinite grammar when certain base sequences (i.e. the simplest possible sequences for a specific grammatical pathway) are put in the test set, thereby excluding them from the training set. For example, in the Reber grammar (appendix A), an example of a base sequence would be *BTXSE*. If this sequence was excluded

from the training set, it would not only prevent the model from being able predict that sequence itself, but may also prevent it from predicting more complex sequences that contain the base sequence in question.

The consequences of the sparse data problem may be that the model's performance is unfairly degraded simply because it isn't seeing a true representation of the grammar. Therefore, in order to counter the effects of this problem the training set should contain all the base sequences from the grammar, as well as the randomly generated sequences. This approach ensured that the model's performance could be accurately gauged during development, without experimental bias introduced by sparse data.

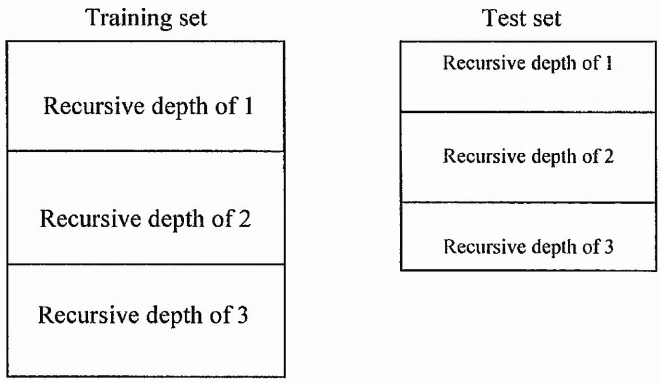


Fig 3.4 – Training and test sets artificially segregated to proportionally represent all levels up to a specific recursive depth.

In order to further enhance the scientific rigor of the experiment it was decided that the training and test sets should be segregated in accordance with the recursive depth of the sequences (fig 3.4). Recursive depth is a term used to denote the number of sequentially repeating symbols generated by a recursive state in the grammar (discussed in section 3.3.1). Segregation based on recursive depth would involve ensuring that the training and test sets contained equal numbers of sequences for each level of recursive depth up to the maximum depth denoted for that experiment. This

would allow tighter experimental control over the training data and would allow the effect of recursive depth could be independently investigated.

3.3.3.2 Experimental parameters

The experiments were conducted using SRSOMs of sizes 20 × 20, 10 × 10 and 5 × 5. It was intended that models within this range of sizes would be large enough to learn the problem, but not so large that they could memorize the grammar. In order to prevent the initial start weights from influencing the results, the same randomly generated weight sets were used for models of each respective size. The models were trained over 1000 epochs using a linearly decreasing learning rate starting from 0.1. However, training was immediately terminated for any model that reached 100% accuracy on the test set. During the training process the model's performance was measured every 10 epochs. In order to gain a measure of average performance each experiment was repeated ten times and the average model accuracy was calculated. The kernel width parameter σ, used in the neighbourhood function, was initially set to half the width of the map (i.e. for a 10 × 10 model σ would be 5). These training parameters were selected using both the recommendation of Kohonen (2001) and the results from previous experiments.

As detailed in section 3.3.3.1, these experiments employed separate randomly generated training and test sets that were segregated according to recursive depth. The models were trained on sequences with a maximum recursive depth of six, but were also tested on set of sequences with a maximum recursive depth of eight and twelve respectively. Testing the SRSOM on sequences with up to twice the recursive depth

encountered during training provided an insight into the models level of generalization.

A side effect of using recursive depth segregation was that it was statistically very hard to generate large training sets. The reason for this is that for lower recursive depths there are less possible sequences for each specific grammatical pathway than for higher recursive depths. Therefore, while the grammar is technically infinite even at a recursive depth of zero (i.e. even without recursive states it still contains the *XVP* loop), it becomes increasingly difficult to randomly generate unique sequences. Consequently the following experiments used training sets varying from 38 to 47 sequences and corresponding test sets from 10 to 14 sequences (sizes were dependant upon recursive depth). These smaller sets provided the model with an optionally proportional representation of the grammar, while ensuring that no base sequences were excluded from the training set.

3.3.3.3 Experimental results

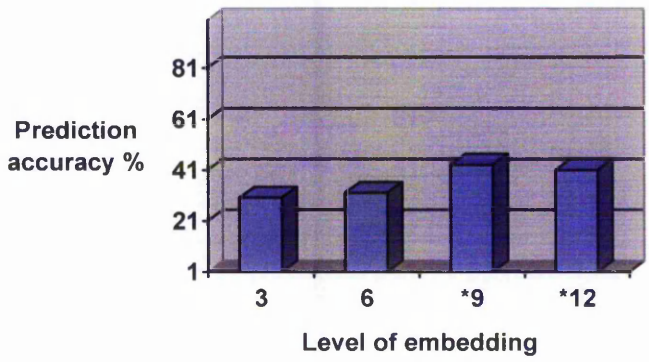


*Fig 3.5 – Results from 4 test sets for a 5 ×5 SRSOM trained on the Reber grammar. The levels of embedding shown with a * denote recursive depths beyond those encountered during training.*

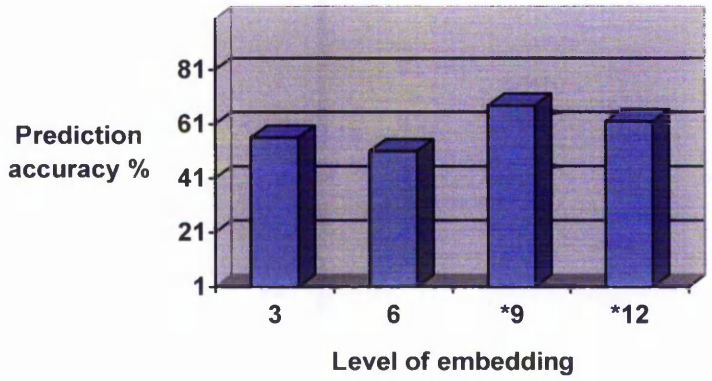


*Fig 3.6 – Results from 4 test sets for a 10 ×10 SRSOM trained on the Reber grammar. The levels of embedding shown with a * denote recursive depths beyond those encountered during training.*

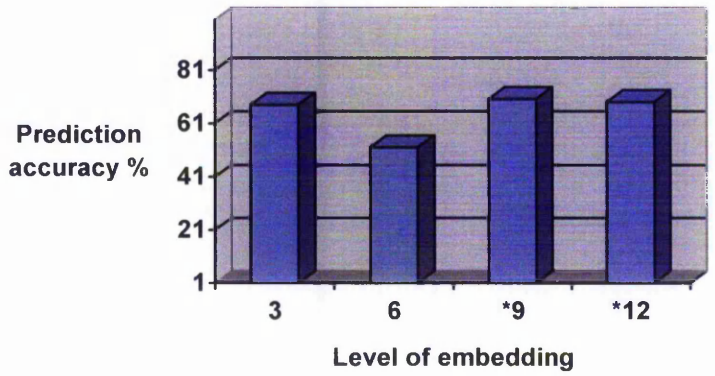


*Fig 3.7 – Results from 4 test sets for a 20 ×20 SRSOM trained on the Reber grammar. The levels of embedding shown with a * denote recursive depths beyond those encountered during training.*

3.3.3.4 Analysis of experimental results

The results in figs. 3.5 - 3.7 show that the SRSOM is able to achieve up to 70% accuracy at predicting the Reber grammar. The 20 × 20 model appears to have learnt the best representation of the grammar, with prediction accuracy marginally above that of the 10 × 10 model. However, performance for the 5 × 5 model barely exceeded 40%, suggesting that 25 neurons are insufficient for this task.

Surprisingly, these results don't appear to show the expected inverse correlation between performance and the recursive depth of the grammar. It was expected that performance would decrease when the SRSOM was tested on sequences with a recursive depth greater than those encountered during training. Sequences containing a higher recursive depth should only be predictable if the model had successfully learnt the rules governing the recursive states in the grammar. Therefore the model's apparent increase in prediction accuracy for higher recursive depths suggests that it may have successfully learnt the rules for recursive states (despite apparently not being able to properly learn the other grammatical rules).

The SRSOMs best prediction accuracy of 70% is comparatively less than the SRN which achieved 100% performance on a similar task (Cleeremans, 1989). However, rigorous analysis proved that only two out of 90 SRNs became perfect grammar recognizers (Sharkey *et al,* 2000) in a similar experiment. Furthermore, experiments also show that the SRN, as with other dynamic recurrent models, is unable to generalize to sequences with a recursive depth significantly higher than encountered in the training set. Thus while the results in figs 3.5 - 3.7 suggest that the SRSOM is unable to properly learn the Reber grammar, the lack of an inverse correlation between performance and recursive depth suggest that the model may capable of learning recursive states.

In order to further analyse the SRSOMs performance, a state analysis was conducted to determine how the activations of winning neurons corresponded to the states in the grammar. Fig 3.8 shows the state diagram for a 10 × 10 model trained on the Reber grammar using a randomly generated training set with a maximum recursive depth of

six. The model was trained on 39 randomly generated sequences that were segregated according to recursive depth and included all base sequences. All training parameters were identical to previous experiments.

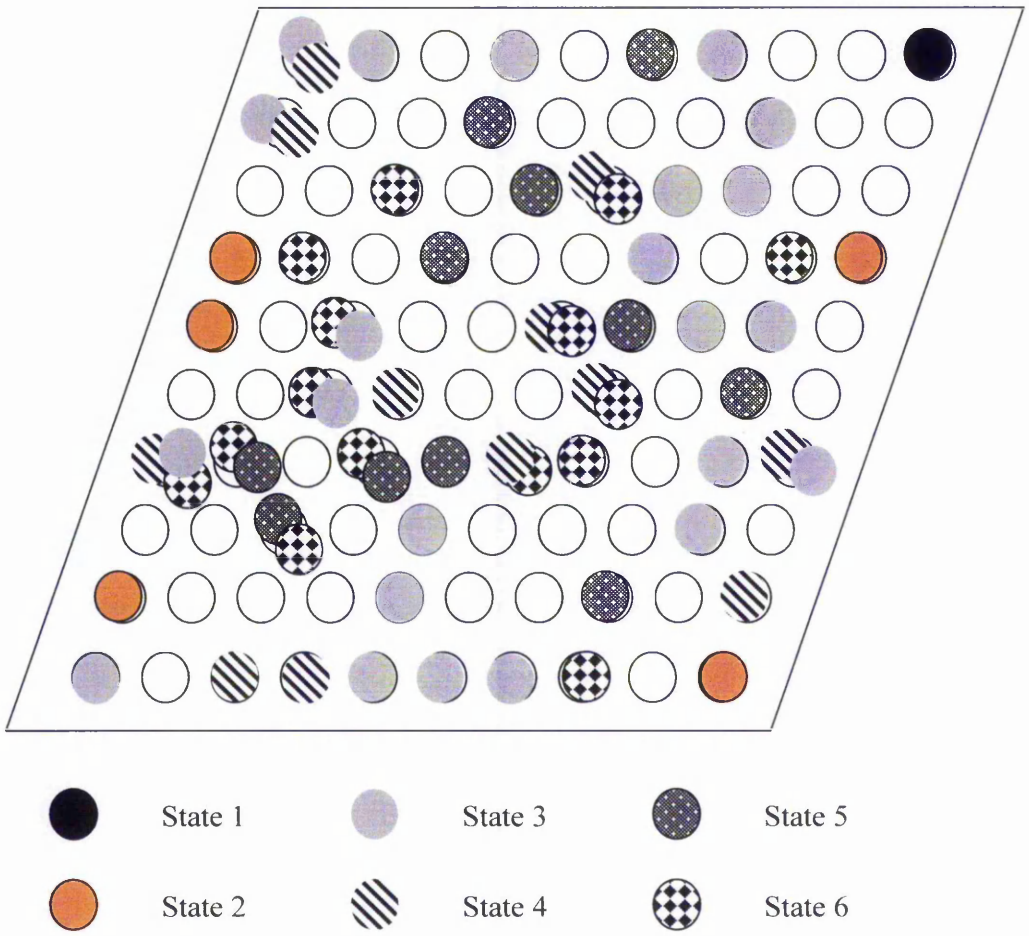


Fig 3.8 – State diagram showing the winning neurons for each state in the Reber grammar for a 10 × 10 SRSOM.

The state diagram is derived from the location of all the possible input symbols for each state in the grammar. For example state 2 is shown if a *T* symbol follows state 1 or an *S* symbol follows state 2. Therefore the locations of the winning neurons for all occurrences of those *T* and *S* symbols constitute state 2. Fig 3.7 shows that, with the exception of state one (which only consists of the '*B*' symbol), the winning neurons for all of the states are spread out over the map. For example while state three

comprises of three input symbols (*P*, *X* and *T*), there are 22 separate winning neurons for symbols corresponding to this state. This diffuse pattern of neuron activations suggests that the SRSOM has memorized separate fragments, rather than learnt rules.

This fragmentation effect poses a problem for the SRSOM's prediction algorithm. It was originally hypothesised that the self-organizing process would result in input symbols being clustered according to their respective states. Consequently, the prediction algorithm operates by finding the two neurons whose context vector is a best match to the location of the current winning neuron (i.e. given the current winning neuron, which two neurons are the most likely next winners?). This approach to prediction operates on the basis that the neurons corresponding to the next grammatical symbols *can* be predicted given the current winner. However, if either of the neurons corresponding to these next symbols has a context vector that doesn't match the location of the current winning neuron, then that next symbol in question cannot be predicted (i.e. if induced knowledge is isolated to each sequence fragment then it cannot be generalized to other fragments). The consequence of this inability to generalize is that the SRSOM would only be able to correctly predict the symbols in a sequence if that particular sequence was part of the training set.

This memorization effect can be further illustrated by conducting a simple experiment that involves testing the model used to produce fig 3.8 using sequence 1 from table 3.8. This particular sequence was not part of the training set, although it is similar to other sequences that were included in the training set. Therefore, while the SRSOM has been trained on all the symbols and grammatical constructs comprising sequence 1, it has not encountered them in that specific order.

#	Winning neurons for symbols in sequence
1	B P V P X V P X T V V E 100 1 60 21 20 72 58 84 4 85 14 85
2	B P V P X T V V E 100 1 60 21 20 61 48 64 46
3	B T S S X S E 100 2 91 7 94 4 85
4	B T S S S S S X X T V P X V P X V P S E 100 2 91 7 97 6 97 10 83 67 44 65 25 37 54 85 14 65 25 36
5	B P V P X V P S E 100 1 60 21 20 72 58 87 5

Table 3.8 – Symbols and corresponding winning neurons from five selected sequences processed by the SRSOM. Sequences 2,3,4 and 5 were included in the training set, but sequence 1 was excluded from training.

Table 3.8 shows that when the SRSOM encounters a symbol in a context that was not seen during training it selects a *best guess* winning neuron, which may have been part of a completely unrelated sequence. For example, when the 9th symbol, *T*, in sequence 1 was encountered the SRSOM selected winning neuron number 4, which corresponds to a state 6 '*S*' symbol from sequence 3. This re-use of inappropriate winning neurons is also shown by the 11th symbol '*V*' in sequence 1, which selects neuron number 14. However, that neuron is actually the winning neuron for a state 5 '*V*' symbol from sequence 4. Not only has the SRSOM selected inappropriate winning neurons from other sequences, but it even selects neuron number 85 twice in order to represent the completely different symbols '*V*' and '*E*' in sequence 1.

While the experiment used to produce the sequences in table 3.8 does show that the SRSOM is unable to generalize appropriately, it also reveals a potentially beneficial

aspect of the model's behaviour. The SRSOM appears to represent recursive states by alternating between two regions of the map. This is illustrated in sequence 4, which shows the model alternatively selecting neurons near 7 and neuron 97 as it processes the state 2 recursive symbol '*S*' . This effect has also been observed in other sequences in the training set and is reflected by the results shown in figs 3.5 - 3.7, which show comparatively higher performance on sequences of a greater recursive depth that those encountered during training.

In conclusion, because the SRSOM doesn't represent states as one zone of the map, it is unable to generalize knowledge learnt in one sequence to other sequences. The model is effectively acting as an associative memory and is simply memorizing the training set. It is unable to generalize appropriately to novel sequences, except in the case of recursive states. Consequently the SRSOM is unsuitable, in its current form, to language acquisition because any potential model must be able to generalize in order to solve the projection problem (Baker, 1979).

4 – STORM

This chapter will identify a fundamental limitation of the SRSOM model that explains why it is unable to learn the underlying rules of the Reber grammar. Following these limitations, a discussion of memory-rule based linguistical models will be given. It will be argued that a memory-rule approach to grammar induction may allow the SRSOM to overcome its memorization limitation and therefore allow it to learn the rules of the grammar.

An enhanced model will then be proposed that is able to learn grammatical rules that allow the model to generalize its learnt behaviour to novel sequences. Following a detailed discussion of this model, a set of experiments will be presented that quantify the model's ability to learn the Reber grammar.

4.1 Limitations of the SRSOM – The need for state

Chapter three concluded with a critical analysis of the SRSOM model. A state diagram was used to show that, in the SRSOM, grammatical states were not clustered into specific regions, but were spread out over the map. Because the model does not perform state-based clustering, it is unable to correlate information learnt from one sequence to that of a functionally-similar sequence. The limitation of this becomes apparent when considering functional-equivalence theory (Hopcroft and Ullman, 1979). This asserts that two states are functionally equivalent if, for all future inputs, their outputs are identical.

Thus, according to functional-equivalence theory, a state is distinguished by the continuity of its outputs rather than by any function of its inputs. This definition implies that the input sequences needed to arrive at any given state need not be related to each other (i.e. their contexts have no common factor). Instead, the state is defined by the fact that each input sequence has an identical output sequence (i.e. the part of the sequence following the state in question). Therefore, if a model is to construct states, it must bind the input symbols that define entry to a state such that for whichever of these state-input symbols occur, the model reaches an identical output state. In the case of a SOM based model, such behaviour may be characterized by activating the same winning neuron for all of the input symbols for a specific state (table 4.1).

#	Input sequence:	Model's representation of input sequence after training:
1	**B T X S E**	**B (T:*STATE 2*) X S E**
2	**BT S X S E**	**B (T:*STATE 2*) (S:*STATE 2*) X S E**
3	**B T X X V P S E**	**B (T:*STATE 2*) X X V P S E**
4	*B T S X X V P S E*	*B (T:STATE 2) (S:STATE 2) X X V P S E*

Table 4.1 – Example illustrating the need for state in order to generalize. If a model is able to learn a representation of state 2 from sequences 1 and 2, then it can learn sequence 3 in a manner that allows it to correctly generalize to sequence 4.

The results shown in chapter 3 demonstrate that the SRSOM model could not achieve such a state-based representation. In the SRSOM each sequence is effectively memorized separately, thus preventing the SRSOM from generalizing to novel sequences. For example, with respect to the sequences shown in table 4.1, the SRSOM would need to be trained on all four sequences in order to correctly learn them all. Because the SRSOM is unable to learn state 2, the third symbol, 'X' in

sequence 3 would activate a different winning neuron to that activated by the 'X' symbol in sequence 2. Therefore, the model would be unable to generalize its knowledge from sequence 3 to sequence 4, requiring that the latter be explicitly included in its training set. This is a profound limitation which implies that in order to be able to learn the grammar all of the symbols must be seen in all possible contexts. As discussed in (Hadley and Cardei, 1999), such a positionally intensive training requirement is cognitively unrealistic.

This discussion of the limitation of the SRSOM implies that any model designed to learn the rules of a grammar must be capable of representing input symbols via state. However, conventional recurrent neural networks, such as the SRSOM, operate using only a representation of the past-context and the current input symbol. This poses a problem, because by themselves the past context and current input simply don't provide sufficient information to unambiguously determine the appropriate state purely from the input data. As previously explained, a state is defined by the continuity of its outputs. The input symbol sequences that lead to a particular state, along with their associated contexts, have potentially no information that could be used to allow the model to attract the sequences towards a common region of its state space.

This premise is illustrated by the FSM for the Reber grammar (see diagram in appendix A), which shows both the grammatical states and their corresponding input symbols. For example, state 4 has the two input symbols X and P and the two output symbols X and S. However, without further information by which to identify state 4, the SRSOM's learning algorithm will separately store representations for the state 4

input symbols X and P in completely unrelated areas of the map. Consequently this disparity will result in duplicate representations for each of the output symbols X and S. Thus the choice of winning neuron for the state 4 output symbol X will depend on whether it was preceded by an X or by a P. This fragmentation effect will be further compounded by separation of input symbols at every state encountered in the grammar.

4.2 Memory-rule based models

Many leading linguists, such as Pinker (2000) and Marcus (2000), have theorized that language acquisition, as well as other aspects of cognition, can be explained using a memory-rule based model. This theory proposes that cognition uses two separate mechanisms that work together to form memory. Such a dual-mechanism approach is supported by neuro-biological research, which suggests that human memory operates using a declarative fact-based system and a procedural skill-based system (Cohen and Squire, 1980). In this memory-rule based theory, rote memorization is used to learn individual exemplars, while a rule-based mechanism operates to override the original memorizations in order to produce behaviour specific to a category. This memory-rule theory of cognition is commonly explained in the context of the acquisition of the English past tense (Pinker, 2000). Accounting for children's over-regularizations during the process of learning regular and irregular verbs constitutes a well-known battlefield for competing linguistic theories. Both Pinker (2000) and Marcus (2000) propose that irregular verbs are learnt via rote-memorization, while regular verbs are produced by a rule. The evidence for this rule-based behaviour is cited as the over-regularization errors produced when children incorrectly apply the past tense rule to irregular verbs (e.g. *runned* instead of *ran*).

As established in the previous section, in order to learn the grammatical rules a model must be capable of identifying and representing grammatical states, in accordance with functional-equivalence theory. However, it was also shown that the SRSOM cannot derive such states using only a representation of the past-context and current input. This problem can be overcome by treating the neural network as a memory rule-based model, in which the future-context of memorized sequences is compared against the future-context of the current input sequence. Therefore, instead of trying to form states using only a representation of the past-context and the current input, as in conventional dynamic connectionist models, a memory-rule based grammar induction model would use regularities in the future-context of sequences to identify states. Thus such a model would learn via *similarity of behaviour*, rather than *similarity of form.* This is illustrated in table 4.2, which shows three sets of example sequences from the Reber grammar.

#	Sequence:	State:
1	B *T* **X S E**	2
2	B T *S* **X S E**	2
3	B T *X* **S E**	4
4	B P V *P* **S E**	4

Table 4.2 – Example of how the future-context of sequences can be used to identify states in the grammar. Sequences 1 and 2 can be used to identify the state 2 input symbols, whilst sequences 3 and 4 can be used to identify the state 4 input symbols.

As shown in table 4.2, the state 2 input symbols 'T' and 'S' are related to each other by their common future-context 'X S E'. State 4 is an even better example of this approach to state-identification, because the symbols 'X' and 'P' that constitute state 4 have completely unrelated past-contexts. Therefore, the only common feature that

can be used to identify state 4 is the identical future-context 'SE' for both of the state 4 input symbols 'X' and 'P'.

4.3 STORM (Spatio-Temporal self-Organizing Recurrent Map)

STORM is a memory-rule based connectionist model. It extends the SRSOM by using lateral interconnections between neurons on the map (fig 4.1). As with the original SRSOM, a recurrent feedback mechanism enables the model to act as a temporal associative memory. This allows the model to initially produce localist-based memorizations of input sequences. The model's rule-based mechanism then exploits the similarities between the future-context of memorized sequences and the future-context of the current input sequence. These similarities drive a temporal-Hebbian learning mechanism that uses the lateral connections to bind functionally-related neurons together, effectively overriding the original memorizations and producing state based rules. The next two sections will discuss the model's memorization and rule-construction mechanisms, respectively.

4.3.1 STORM's memorization mechanism

STORM's memorization mechanism allows it to initially learn input sequences in exactly the same manner as for the SRSOM. Its recurrency mechanism feeds back the location of the previous winning neuron on the map, which has the effect of mapping similar elements of different sequences to different winning neurons (fig 4.1).

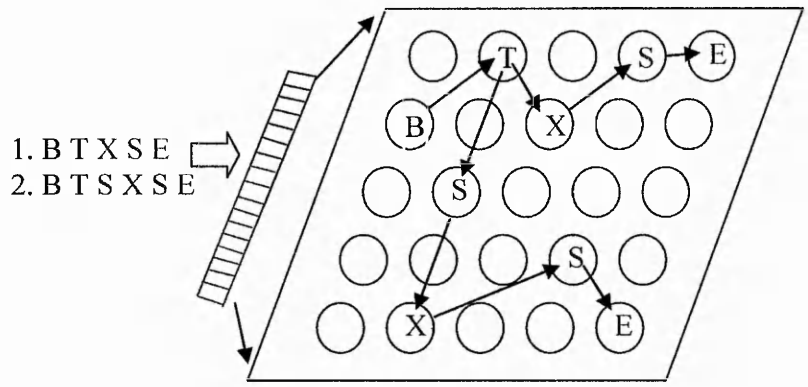


Fig. 4.1 – The winning neurons for two memorized sequences that end with the same sub-sequence 'XSE'.

4.3.1 STORM's rule-based construction mechanism

The model's location-based recurrency representation and localist architecture provide it with a very important ability. Unlike conventional connectionist models, the sequences learnt by STORM can be extracted in reverse order. This makes it possible to start with the last element in an input sequence and work backwards to find the winning neuron corresponding to the previous symbol in the sequence. This is achieved in a similar manner to the operation of the SRSOM's prediction algorithm, discussed in section 2.3.2. Because each neuron's weight vector consists of a symbol and context vector containing the coordinates of the previous winning neuron, it is possible to examine a neuron's context-vector to find the previous winning neuron in a sequence.

STORM's rule-based construction mechanism exploits this ability to transverse memorized sequences in both directions, allowing it to identify sequences whose future-contexts match those of the current input sequence. As explained on page 74, symbols that share the same future-context represent the input-symbols to the same

states. This is illustrated in figure 4.2, which shows that the 'S' and 'T' input symbols to state 2 can be identified by their common future context 'X S E'.

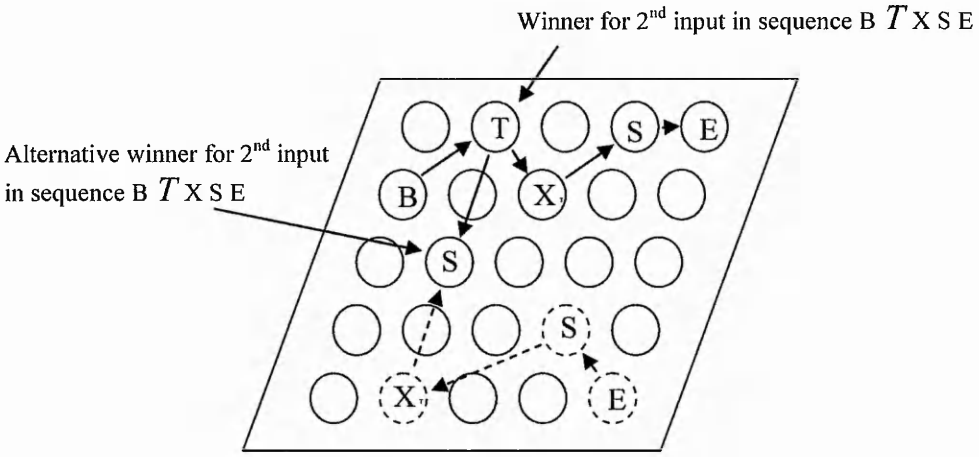


Fig. 4.2 – Illustration of the alternative winning neuron selection algorithm tracing back through the stored sequence BTSXSE. The neuron representing the third symbol, 'S', is the alternative winner with respect to the second symbol, 'T' in the sequence BTXSE. This is because both the 'T' and the 'S' have the same future context (XSE).

When an input is presented to the model, the winning neuron selection algorithm finds the best matching neuron, exactly as in the SRSOM model (section 2.2.1). However, STORM also traces backwards through the future-context of the input sequence (i.e. starting at the end of the input sequence and working back towards the current input) in order to find an *alternative* winning neuron (fig 4.2). Thus, while the winning neuron represents the best match for the current input symbol and context vector, the alternative winning neuron represents a possible functionally-equivalent input symbol in a sequence already memorized by the model.

STORM's rule-based construction mechanism uses these similarities in future-context to drive a temporal-Hebbian learning mechanism that operates on the lateral connections between neurons (discussed in more detail in section 4.3.2). This

mechanism uses the lateral connections to bind functionally-related neurons together into representations of *state*. This is achieved by strengthening the connections between neurons representing symbols with consistently identical future-contexts, thus allowing the model to build up relationships between functionally equivalent neurons.

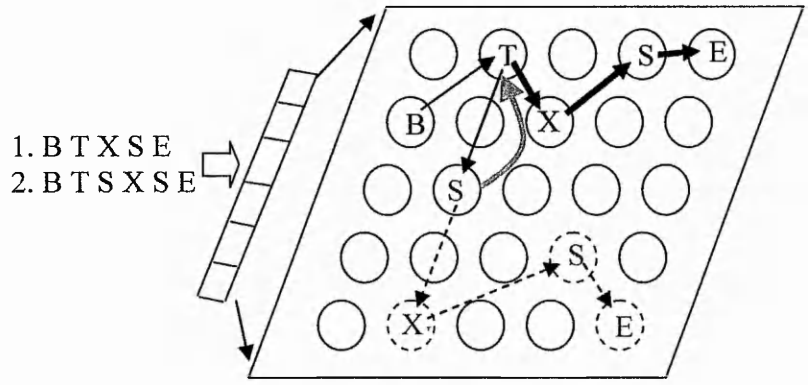


Fig. 4.3 – Functional override in winning-neuron selection algorithm. The functional relationship (shown in grey) between the third symbol, 'S', in sequence 2 and the second symbol, 'T', in sequence 1 forces the model to process the remaining elements in the second sequence (namely 'XSE') using the same winning neurons as for the first sequence.

Once the strength of these lateral connections exceeds a predetermined threshold, they override the recurrency mechanism. This forces the model to use a single representation for the future inputs in the sequence, rather than the original two representations (fig 4.3). This is achieved by using the lateral connections to mark one of the functionally-related neurons as the master and the other as a slave. Therefore, regardless of which of these neurons is the actual winner, the recurrency mechanism feeds-back a representation of the master winner. The criteria for determining which functionally-related neuron is the master and which is the slave is discussed in section 4.3.2.

The result of this override mechanism is that the remaining elements in the input sequence are processed as if the master neuron had been selected, rather than the actual winning neuron. The purpose of forcing the model to use this single representation for the future sequence elements is that knowledge from other sequences, which involves the functionally-related symbols in question, can then be generalized to apply to both symbols (discussed further in section 4.4).

The model's alternative winner selection algorithm conforms to the SOM's winner-take-all philosophy by choosing the alternative winning neuron whose future-context is the best match to that of the current input sequence. Given that tracing back through the future-context may identify multiple alternative winners, the criteria of best matching winner classifies the best matching sequence stored in the model as the winner (i.e. the sequence whose winning neurons are the best match to the symbols in the future-context in question). Furthermore, the lateral connections between the winner and the alternative winner are only strengthened if the future-context for the alternative winner is a better match than the future-context for the winner itself. Thus, the model has a preference for always using the dominant sequence and it will use the temporal-Hebbian learning mechanism to re-wire its internal pathways in order to re-use the best matching representation of a sequence where ever possible.

4.3.2 Operation of the temporal Hebbian learning mechanism

As previously mentioned, STORM exploits similarities between the future-context of memorized sequences and that of input sequences in order to drive a temporal-Hebbian learning mechanism. By using the lateral inter-connections between neurons, this mechanism is able to bind functionally-related neurons together into states. These

inter-connections are constructed so that the neuron whose future-context trace is the best match to the future-context in the current input sequence is denoted the master. The other neuron with the weaker future-context trace becomes the slave (figure 4.4). Thus after state construction, a neuron with a lateral connection to a master will override the recurrency mechanism by using the location of the master neuron as context, rather than the slave neuron itself.

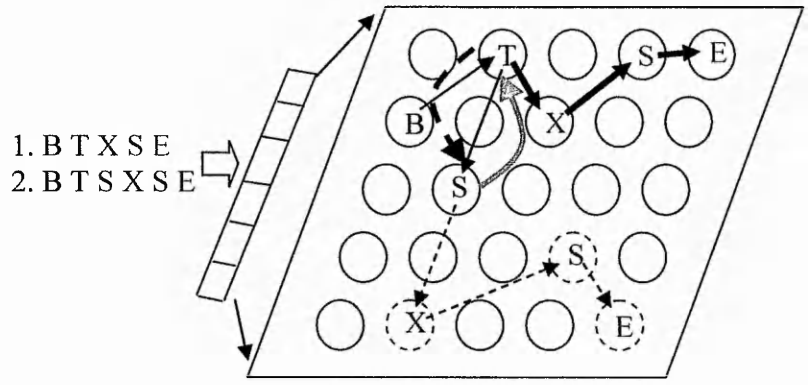


Fig. 4.4 –Master lateral connection from the 'S' neuron to the 'T' neuron (light arrow) and a slave lateral connection from the 'T' neuron to the 'S' neuron (dotted dark arrow).

STORM is not a monotonic learning system. In order to be able to respond to statistical relationships in the input data, the model must be able to both construct and also break-down states. This ability is necessary because occasionally *phantom* future-context traces may be detected by the rule-construction algorithm. Such traces may arise from either statistical anomalies or may be the result of obsolete sub-sequences whose winning neurons have shifted position during the training process. Without the ability to break-down functional-relationships these *phantom* future-context traces would interfere with the learning process by masking the real functional-relationships.

The temporal-Hebbian learning mechanism both constructs and breaks-down states by applying a positive and a negative learning rate to the lateral connections. While the positive learning rate is applied with respect to regularities in the future-context, the negative learning rate is continually applied to all lateral connections and acts as a global decay (Figure 4.5). Upon presentation of each input symbol to the model, all lateral connections decay by a pre-determined value (typically in the order of 0.005). The effect of this decay is that obsolete functional-relationships between neurons (i.e. relationships that are no longer reinforced via a positive learning rate) are slowly broken down over time.

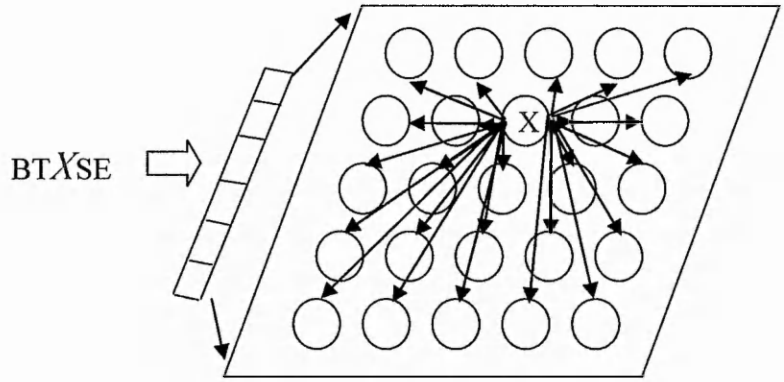


Fig. 4.5 –Illustration of the negative learning rate being applied to all of the lateral connections for the neuron representing the symbol 'X'.

The model's negative learning rate is complemented by a positive learning rate, which is used in the construction of functional-relationships between neurons (Figure 4.6). However, due to the operation of the model's state construction algorithm, the application of this positive learning rate is slightly more complex than its description suggests. As previously discussed, similarities between the future-context of memorized sequences and that of input sequences, are used to construct states, forcing the model to use only one future-context representation rather than two (figure 4.4). However, by forcing the model to share a single representation for the future-context

of both state-inputs, the very criteria used to construct the functional-relationship (i.e. two identical future-contexts) is removed. Therefore, once a state is formed the criterion used to identify it disappears. This subsequently prevents the temporal-Hebbian learning mechanism from identifying and reinforcing the lateral connections that constitute the functional-relationship. Without such reinforcement, the effects of the negative learning rate slowly break down the functional-relationship between the state-neurons, until eventually the state collapses and the process of state formation must begin all over again.

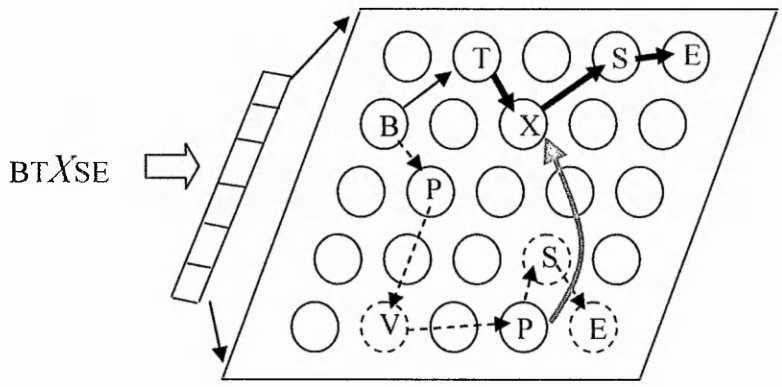


Fig. 4.6 – Application of positive lateral learning rate. When the 'X' from the 1st sequence BTXSE is input, the lateral connection belonging to the alternative winning neuron (the 2nd P in the memorized 2nd sequence BPVPSE) is enhanced with respect to the winning neuron for the current 'X' input from the 1st sequence.

This conundrum is resolved by constructing functional-relationships using an approach called mutual-activation-reinforcement. In this approach winning neurons don't enhance their own lateral connections. Instead they enhance the lateral connections belonging to the alternative winning neuron, identified via an identical future-context. For example, in figure 4.7 the winning neuron for the 4th symbol, 'P', in sequence 2 will enhance the lateral connections belonging to the winning neuron for the 3rd symbol, 'X', from sequence 1. Conversely, the neuron representing the 'X' symbol in sequence 1 would be responsible for enhancing the lateral connection to the

winning neuron for the 4th symbol, 'P', in sequence 2. This mutual-activation-reinforcement will only occur if the winning neurons for the symbols in question are activated in sequences with identical future-contexts. In this way, a two-way functional-relationship between the two neurons is built up. The strength of the lateral inter-connections is continually enhanced as training proceeds until a point is reached where the recurrency mechanism is overridden to form state. This point is determined by an empirically defined threshold value.

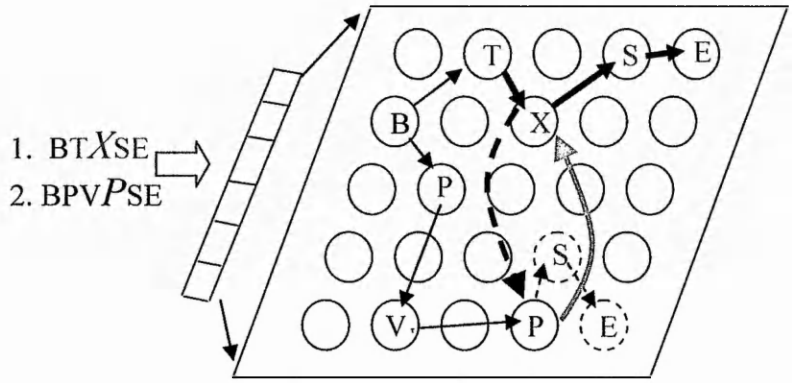


Fig. 4.7 – Mutual-activation-reinforcement. The winning neuron for the 'X' input in sequence 1 enhances the lateral weights belonging to its alternative winner (shown by a light grey arrow). Conversely, the winning neuron for the 'P' in sequence 2 enhances the lateral weights belonging to its alternative winner (shown with a dotted dark arrow).

Once a neuron's lateral weights have exceeded the empirically determined threshold (i.e. the given point at which the recurrency mechanism is overridden) a two-way functional-relationship can be assumed to exist between the neuron in question and the neuron referenced by the active lateral weight. The existence of this relationship is validated by the fact that winning neurons cannot enhance their own lateral weights. Therefore, any active lateral connections could only have been created by a functionally-related neuron. Consequently, at this stage in the state-construction process, it's no longer necessary for both winning neurons to have to identify each

other via future-context matches. Once active lateral connections have been constructed, each winning neuron can directly enhance the other neuron's lateral weights. The only criterion for this enhancement is that the future-context of the current input sequence matches that of the winning neuron itself (i.e. the input symbol in the current sentence isn't an exception to the rule represented by the lateral weight we want to enhance).

Therefore, using the mutual-activation-reinforcement approach discussed above, the assumed functionally-related neuron's lateral weights can be enhanced without the need to identify the alternative winning neuron via identical future-contexts. This overcomes the problem caused by the eventual decay of connections between the slave neuron and the master winning neuron (shown by the dashed 'SE' neurons in fig. 4.7). Thus, while regularities in the future-context are used to initially construct states, once formed the state neurons continually reinforce their relationship with each other, as long as all the neurons that constitute a particular state continue to be regularly activated in the appropriate context.

4.4 Learning via functional generalization

The previous two sections have discussed the mechanics of STORM's state-construction algorithm, but neither has shown the advantages of forming a state-based representation. This section will discuss how STORM's ability to form state allows it to generalize learnt knowledge to unseen sequences and consequently learn from a sparse training set.

While a state is constructed based on similarities in future context, there may be cases where the future context, for the respective input symbols that constitute the state, is dissimilar. For example, the first two sequences in table 4.3 are sufficient for the model to form state 2 (see fold out sheet in Appendix A for the Reber grammar FSM). However, the third sequence in table 4.3 contains the same state 2, but ends with a different sub-sequence (i.e. X X V V E, as opposed to X S E).

#	Symbols in training sequence (winning neurons for symbols)
1	B T X S E (4) (10) (14) (20) (25)
2	B T S X S E (4) (10) (8) (14) (20) (25)
3	B T X X V V E (4) (10) (14) (2) (12) (18) (23)

#	Test sequence (not seen during training)
4	B T S X X V V E (4) (10) (8) (14) (2) (12) (18) (23)

Table 4.3 – Functional generalization example. When trained on the first three sequences, STORM is able to construct a state between the 'T' in sequence 1 and the first 'S' in sequence 2. By generalizing this learnt state to its memorization of sequence 3, STORM is then able to correctly process sequence 4 by activating the same winning neurons for the sub-sequence 'X X V V E' as would be activated in sequence 3.

Once a state has been constructed, the future context in subsequent sequences containing that state will be processed in an identical manner, regardless of whether that future-context is different from that used to initially construct the state 2. To be more specific, the 'T' symbol from sequence 1 will form a state with the first 'S' symbol from sequence 2. This will result in both sequences 1 and 2 sharing the same winning neurons for their final three inputs (X S E). STORM will then be able to generalize this learnt state to its memorization of sequence 3, resulting in the same

winning neurons being activated for the 'X X V V E' in test sequence 4 as are activated in training sequence 3.

As STORM learns specific states, that state-knowledge is reflected in changes to its internal representations of all memorized sequences containing the state in question. This powerful generalization ability reduces the effects of the poverty of stimulus theory, which stipulates that language input is too sparse for a learner to acquire the rules. As discussed in the literature study (section 2.1), the poverty of stimulus theory has been one the central nativist arguments against the feasibility of connectionist language acquisition. STORM's functional generalization abilities preclude the need to learn separate examples of each symbol in different contexts, thus allowing the model to induce a grammar from a sparse data set.

4.5 Experiments

In order to gauge STORM's grammar induction abilities compared with the SRSOM, the model was applied to the same task of predicting the next valid symbols in the Reber grammar. As with the previous SRSOM experiments, separate randomly generated training and test sequences were used to ensure the model performance was assessed by testing it on sequences not encountered during training.

The maximum recursive depth parameter in table 4.4 denotes the highest possible sequential number of recursive symbols in any sequence (i.e. the maximum number of times around a recursive loop, such as the recursive 'S' in the Reber grammar's state 2). By only selecting sequences for the training and test sets with a maximum recursive depth of less than six, it is possible to ensure the model is not tested on

sequences too far beyond the complexity of its training set. The basis of this criterion is the established fact that supervised models such as the SRN become unstable when tested on sequences whose recursive depth is more than three symbols greater than the sequences in its training set (Omlin, 2001). A model size of 10 × 10 neurons was chosen for this experiment because it was believed that this would be sufficiently large enough to allow the learning mechanism to operate, but small enough to prevent the model from just memorizing the grammar.

Parameter:	**Value:**
Number of epochs	1000
Learning rate (linearly decreasing and decremented each epoch)	0.1
Neighbourhood (linearly decreasing and decremented each epoch)	5
Positive / negative temporal Hebbian learning rate	0.5 / 0.005
Lateral activation threshold value	1
Number of training sequences	30
Number of test sequences	10
Maximum recursive depth (RD) of sequences	6
Model size	100 neurons (10 × 10)

Table 4.4 – Experimental parameters for experiment 1.

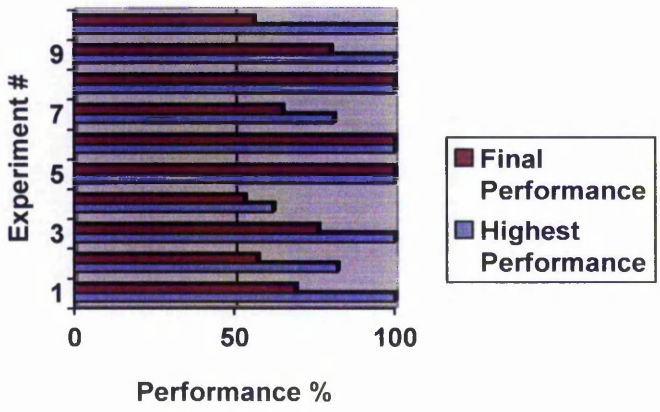


Fig 4.8 – Results from experiment 1 showing the prediction performance on test set 1 for a 10 ×10 STORM model trained on trained set 1.

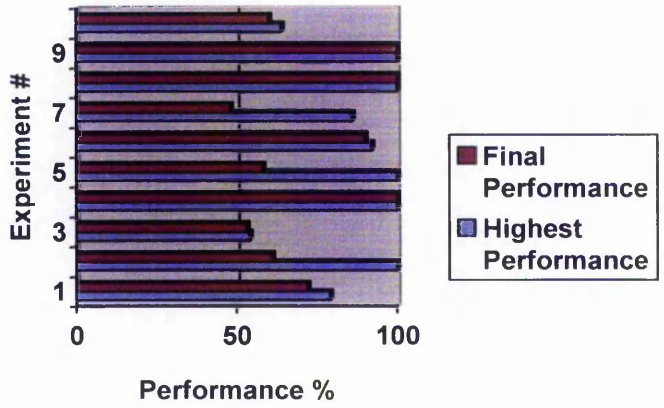


Fig 4.9 – Results from experiment 1 showing the prediction performance on test set 2 for a 10 ×10 STORM model trained on training set 2.

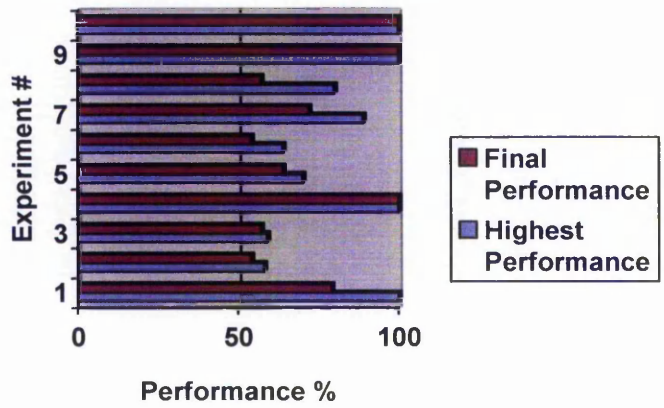


Fig 4.10 – Results from experiment 1 showing the prediction performance on test set 3 for a 10 ×10 STORM model trained on training set number 3.

The results from experiment 1 (figs 4.8 - 4.10) show that between four and seven models became perfect grammar recognizers (100% performance) during training. However, the number of perfect recognizers fell to three, for each set of training/test sets, by the end of training. While these results show that the model is capable of perfectly learning the grammar for multiple training sets, the discrepancy between highest and final performance suggests that the model is unstable. It appears that in some cases optimal representations reached during training are lost by the end of the training process. The exact reason for this is unclear, however in comparison with similar experiments on the SRN (Sharkey *et al,* 2000), the results for STORM are very good because in the SRN experiments only two out of ninety SRNs became perfect grammar recognizers at the end of training.

In order to determine the effect of the model's size on its performance, a second experiment was run that used a model of 25 neurons arranged in a 5 × 5 lattice. This experiment used exactly the same parameters as the first experiment, with the only difference being the model's size.

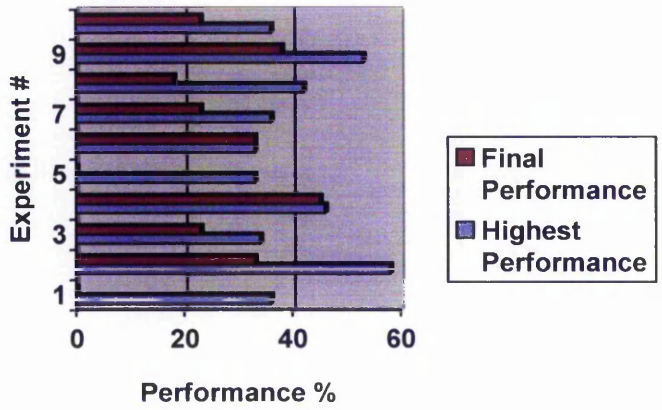


Fig 4.11 – Results from experiment 2 showing the prediction performance on test set 1 for a 5 × 5 STORM model trained on training set 1.

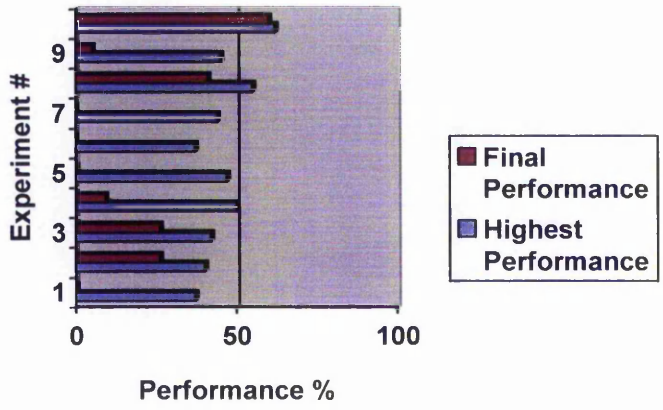


Fig 4.12 – Results from experiment 2 showing the prediction performance on test set 2 for a 5 × 5 STORM model trained on training set 2.

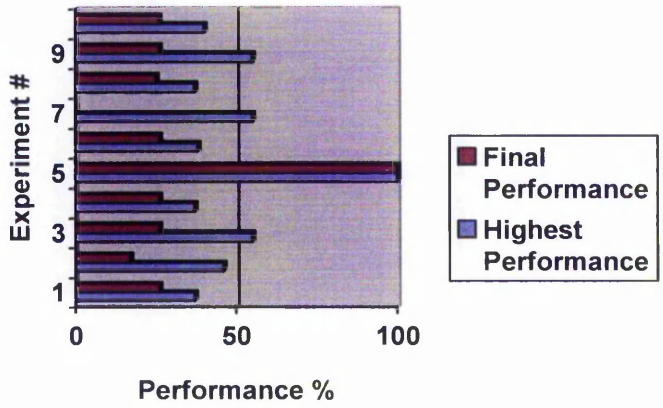


Fig 4.13 – Results from experiment 2 showing the prediction performance on test set 3 for a 5 × 5 STORM model trained on training set 3.

As the results in figs 4.11 – 4.13 show, the smaller 5 × 5 model only managed to achieve 100% performance once in 30 experiments. This suggests that having too few neurons impedes the model's ability to learn the grammar. In order to further investigate the effect of model size on performance, a third set of experiments were run using a model of 15 × 15 neurons. All the parameters were identical to the previous two experiments, except for the model size.

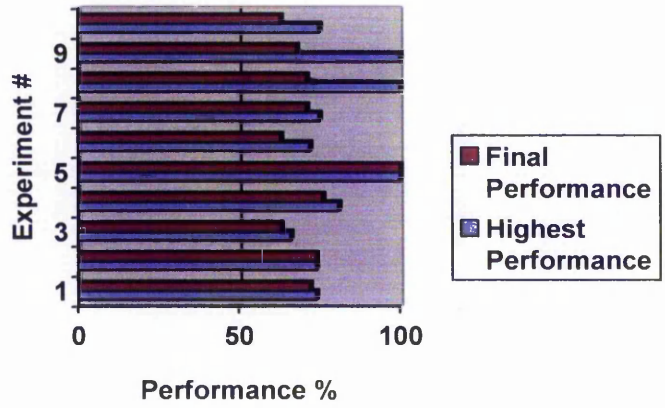


Fig 4.14 – Results from experiment 3 showing the prediction performance on test set 1 for a 15 × 15 STORM model trained on training set 1.

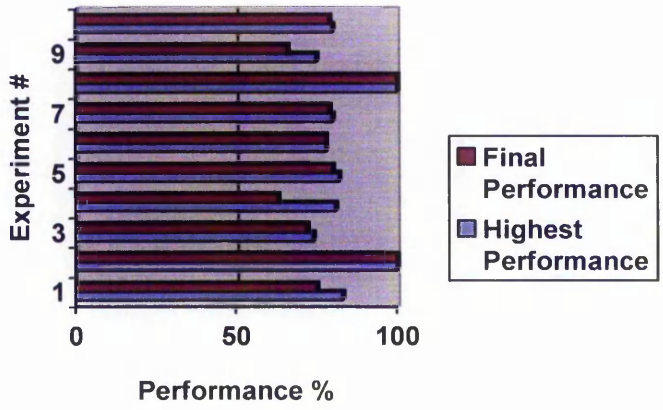


Fig 4.15 – Results from experiment 3 showing the prediction performance on test set 2 for a 15 × 15 STORM model trained on training set 2.

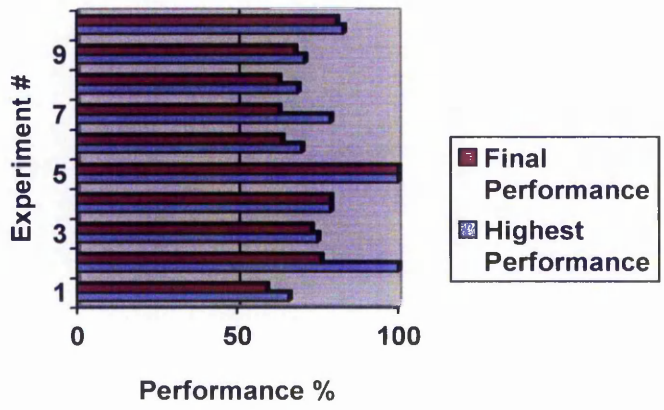


Fig 4.16 – Results from experiment 3 showing the prediction performance on test set 3 for a 15 × 15 STORM model trained on training set 3.

The results in fig 4.14 – 4.16 show that the 15 × 15 model achieved much better performance than the smaller 5 × 5 model. However, rather surprisingly, the 15 × 15 model didn't perform as well as the 10 × 10 model (figs 4.8 – 4.10). In the first experiment, between four and seven of the 10 × 10 models reached 100% performance during training. In contrast, no more than three of the 15 × 15 models reached 100% performance in the third experiment. Given the inferior performance of the smaller 5 × 5 model with respect to the 10×10 model, it was expected that the larger 15 × 15 model would have provided the highest performance out of all the models. A possible explanation for this anomalous result may be that the initial random weights used in experiment three were biased towards a local, rather than a global minima.

In order to investigate the effect of the initial start weights on the model's ability to learn the grammar, an experiment was conducted that varied the start weights, while keeping all other experimental parameters fixed. For this experiment ten models were trained using training set one. Each model had different random initial weights, but the order of the presentation for the training patterns was kept the same by using a fixed seed for the random number generator.

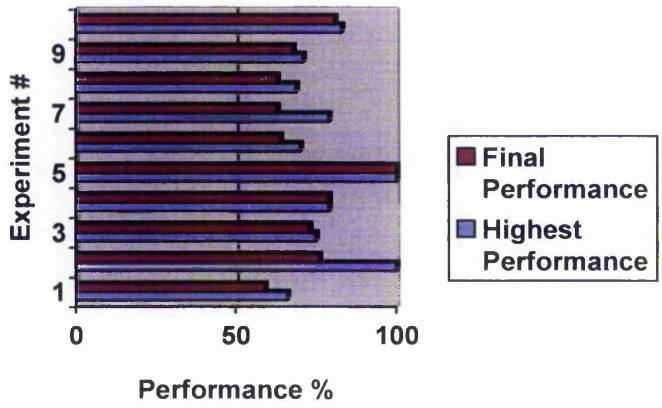


Fig 4.17 – Results from experiment on 15×15 model using training set 1 with random weights, but fixed training sequences presentation order.

The results in fig 4.17 show that the experiment to vary the initial random weights resulted in none of the ten models learning a perfect representation of the grammar. The variation in the results does show that the model is sensitive to the initial start weights and that despite the training sequences being presented in a fixed order, the difference in start weights had an effect on the model's performance. However, it is not clear whether the initial start weights were directly responsible for trapping the model in a local minima or whether they just triggered an instability in the learning algorithm.

4.6 Analysis of STORM

The anomalous results from the experiments on the 15×15 model suggest that there may be unresolved issues involving STORM's learning algorithm. In order to further investigate the learning process, this section of the thesis will provide a full analysis of two models trained on the Reber grammar. The first model to be analysed will be a 10×10 model that achieved 100% prediction performance on training set 1, while the second analysis will involve an identical model that failed to achieve 100% performance. Comparing and contrasting these analyses should provide an insight into the model's anomalous performance and also its internal representation of the grammar.

Table 4.5 shows the corresponding winning neurons for a set of sequences used to test a 10×10 model that achieved 100% prediction performance on the Reber grammar. These test sequences were specifically chosen to include all the simple sequences, as well as a few complex sequences from the Reber grammar. As discussed in section 4.3.1 the proposed advantage of STORM's rule-based learning algorithm is that it allows the model to form a representation of state. Such a state-based representation should allow the model to activate winning neurons in accordance with the current state, as opposed to activating strictly input plus context-dependant winning neurons. The activations in table 4.5 show that STORM's rule-based learning algorithm is indeed forming the intended state-based representation. For example, sequences 1, 6 and 10, which all share the same 'X S E' ending, have identical winning neurons for their last three symbols. Thus despite the variations in context that result from different initial symbols in these sequences, the model correctly reaches an equivalent state when it encounters the final three symbols in each sequence. This state-based

representation is also highlighted by sequences 1, 3, 5, 6, 7, 9 and 10, which all share the same winning neurons for their 'S E' ending, despite massive variations in context. Other states can also be observed inside sequences, such as state 5 (neuron 96) in sequences 3,4,7,8,9 and 11.

#	Input sequence / winning neurons
1	B T X S E 10 4 80 21 91
2	B P V V E 10 1 96 57 91
3	B T X X V P S E 10 4 80 51 96 50 21 91
4	B T X X V V E 10 4 80 51 96 57 91
5	B P V P S E 10 1 96 50 21 91
6	B T S X S E 10 4 100 80 21 91
7	B P T V P S E 10 1 7 96 50 21 91
8	B T X X T V V E 10 4 80 51 7 96 57 91
9	B T X X V P X V P S E 10 4 80 51 96 50 51 96 50 21 91
10	B T S S S S S S S X S E 10 4 100 100 100 100 100 100 100 80 21 91
11	B P V P X T T T T T T T V V E 10 1 96 50 51 7 7 7 7 7 7 7 96 57 91

Table 4.5 – Activations from a 10×10 model that achieved 100% prediction performance on the Reber grammar. The rightmost column shows sequences of input symbols above the winning neurons for the respective symbols.

Sequences 10 and 11 show how this state-based representation allows the model to correctly process sequences of a higher recursive depth than those encountered during training (i.e. the recursive 'S' symbol in sequence 10 will always activate the same winning neuron regardless of how many 'S' symbols there are).The state-based representation learnt by the 100% model is shown graphically in figure 4.18. In order to represent the state-relationships encoded in the lateral connections, figure 4.18

shows the winning neurons after any functional-overrides have been applied. The activations were produced by testing the model on the same set of sequences shown in table 4.5.

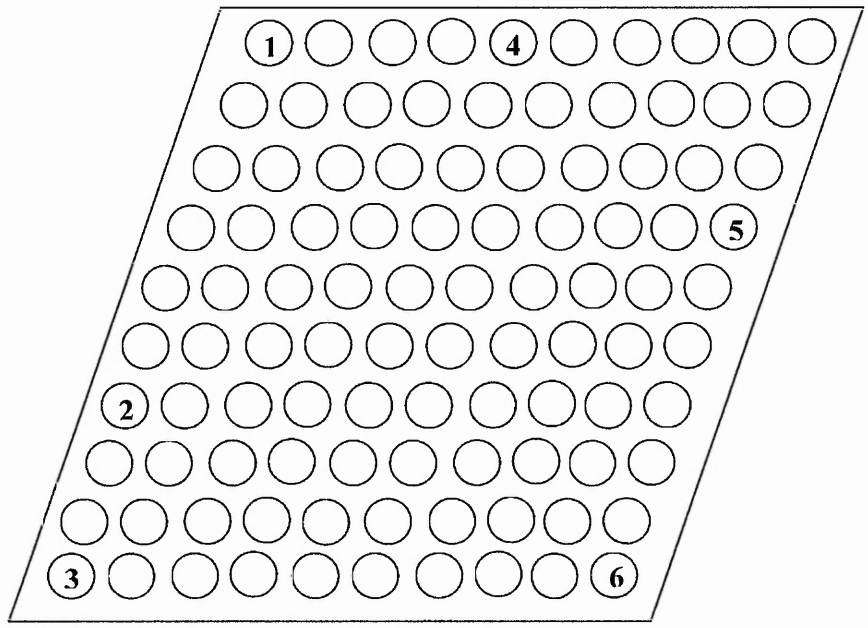


Fig. 4.18 – State diagram for 100% model. The numbered circles represent the winning neurons for the respective grammatical states (shown after functional-overrides have been applied).

The state-diagram in figure 4.18 illustrates the model's perfect representation of the grammar. Despite being tested on eleven sequences with a total of 90 input symbols, the model only uses six neurons (after the functional-override has been applied) in its internal state representations. These six neurons represent the six states in the grammar, proving that the model has learnt a perfect representation of the Reber grammar.

The analysis of the model that successfully learnt the Reber grammar shows that STORM's learning algorithm performs as expected, forming a state-based representation of the grammar. However, as the experiments in section 4.5 show, not

all models are capable of learning a perfect representation of the grammar. In order to investigate the failures, a second analysis was performed on a 10×10 model that achieved less than 100% prediction performance on the Reber grammar. As with the previous analysis, the model was trained on training set 1 and then tested on a set of sequences specifically chosen to show the model's full range of performance on the grammar.

#	Input sequence / winning neurons
1	B T X S E 1 31 90 50 84
2	B P V V E 1 61 4 91 100
3	B T X X V P S E 1 31 90 97 29 54 68 93
4	B T X X V V E 1 31 90 97 29 10 93
5	B P V P S E 1 61 4 91 60 84
6	B T S X S E 1 31 70 86 59 95
7	B P T V P S E 1 61 23 15 63 69 85
8	B T X X T V V E 1 31 90 97 47 28 10 93
9	B T X X V P X V P S E 1 31 90 97 29 54 77 30 64 68 93
10	B T S S S S S S S X S E 1 31 70 75 59 85 59 85 59 96 50 84
11	B P V P X T T T T T T T V V E 1 61 4 91 100 48 37 36 45 36 45 36 17 9 94

Table 4.6 – Activations from a 10 ×10 model that achieved only 74% prediction performance on the grammar. The rightmost column shows sequences of input symbols above the winning neurons for the respective symbols.

The activations in table 4.6 show that the model has not properly learnt the rules of the grammar. For example, although sequences 1, 7 and 9 all end in with the same 'S E' symbols, each sequences activates different winning neurons. This lack of state-based representation can also be observed right at the beginning of the sequences (i.e.

sequences 1, 6 and 10 all have different winning neurons for the 'X' symbol). There are two possible explanations for these results. Either the model has failed to learn any rules at all, or it has learned partial rules that apply only to specific sequences. In the former case, the model would be acting just like the SRSOM from chapter 3 and simply finding the best match for the input and context vectors. As such, the identical winning neurons for the last two symbols in sequences 3 and 9 would occur not because of a rule, but because both the previous winning neurons for both sequences were adjacent to each other.

The second explanation for the results in table 4.6 is that the model in question has learnt partial rules that apply only to specific sequences. In this case the identical winning neurons for the last two symbols in sequences 3 and 9 could be explained by the existence of a rule that applies only to those specific sequences (or more correctly, to two sequences similar to 3 and 9 because neither sequences were part of the training set themselves). If this were the case then this suggests that a failure has occurred early in the learning process, resulting in the formation of erroneous states. Such a situation may have arisen if the model was prevented from finding the basic states (i.e. states two and three) and was therefore forced to construct states five and six based on specific memorized sequences. However, subsequent analysis strongly suggests that the former explanation is the case in point.

The state-diagram in figure 4.19 validates the premise that the 74% model's inadequate performance was due to its lack of a state-based representation. Despite a few clusters, the majority of activations for the six states are scattered throughout the map. As explained in section 4.1, without a state based representation a model is

limited to learning fragmented memorizations of input sequences. Because its knowledge of the grammar is relative to particular contexts rather than states, it is unable to generalize knowledge from one sequence to a sequence in a different context.

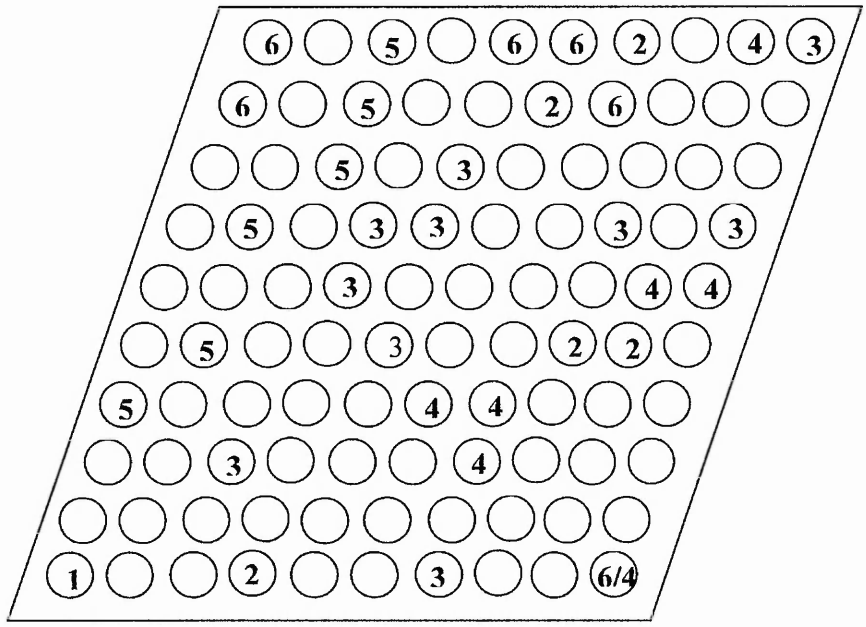


Fig. 4.19 – State diagram for 74% model. The numbered circles represent the winning neurons for the respective grammatical states.

4.7 Conclusions

The SRSOM model's lack of a state-based representation meant that it stored input symbols relative to particular contexts rather than states. This consequently prevented the SRSOM from generalizing information learnt from a symbol in a particular context to equivalent situations in different contexts. The requirement for a state-based representation was clarified by defining state in accordance with functional-equivalence theory. The implication of this theory is that states can only be created by identifying potential input symbol sequences that have matching future-contexts. As such, conventional recurrent neural networks such as the SRSOM, which operate

using only a representation of the past-context, are unable to form an adequate representation of state.

In order to overcome the limitations of the SRSOM, a novel connectionist memory-rule based model was proposed. STORM is able to create a state-based representation of the grammar by exploiting the future-context of sequences, in accordance with functional-equivalence theory. By operating as a memory-rule based system, the model is able to compare the future-context of symbols in an input sequence with the future-context of symbols in memorized sequences. This allows the model to identify functionally-related symbols and bind their associated neurons together using a temporal-Hebbian learning mechanism. This has the effect of constructing states in a bottom-up manner and learning using *similarity of behaviour*, rather than *similarity of form*. Once constructed, these states affect the processing of subsequent input symbols, forcing the model to use a state-based representation rather than distinct representations for each symbol.

Multiple experiments prove that STORM's state-based representation allows several instances of the model to learn a perfect representation of the Reber grammar which, in turn, allows it to generalize beyond its training set. However, these experiments also suggest that there are unidentified issues relating to the learning algorithm which sometimes prevent the model from learning a correct representation of the grammar.

In summary, STORM is a radical connectionist model whose state-based functional learning algorithm allows it to supersede the capabilities of conventional connectionist grammar induction models, such as the SRSOM. By inducing explicit

representations of the rules of a grammar from sparse data, STORM poses a challenge to traditional natavist linguistic theories and further undermines the logical problem of language acquisition (Jackendoff, 2002). Furthermore, STORMs reliance on functional-equivalence theory and its memory-rule based approach to grammar induction, provide a new pathway for connectionist modelling that brings the discipline closer into line with traditional linguistics.

5 Beyond the Reber grammar

Chapter four's presentation of the STORM model used the Reber grammar as a test-bed. However, while this simple regular grammar provides an excellent example of a basic generative grammar, it also lacks some of the important complexities found in natural language. This chapter will discuss the problems posed by more complex grammars, including recovery from over-generalization and the redundancy caused by centre embeddings. Extensions will be proposed to allow STORM to overcome these problems and potentially scale up closer to natural language itself. However, preceding the discussion of complex grammars, this chapter will highlight a limitation that may explain the model's sporadic failure at inducing a perfect representation of the grammar.

5.1 Instability and Causality Loops

During the development of the STORM model, a number of refinements and concessions had to be made in order to turn the original conceptual model into an operational prototype. The temporal-Hebbian learning mechanism in STORM's rule-construction algorithm is one such example of a compromise between the conceptual model and the resulting implementation. In the conceptual model it was envisaged that states could be constructed by encouraging functionally-related neurons to move towards each other on the map. In this way, the original topological SOM would be transformed into a functional SOM, with the neurons representing states located together in clusters. However, initial unpublished experiments established that the neighbourhood function interfered with the rule-construction algorithm, resulting in catastrophic instability. This interference occurred because the neighbourhood

function was attempting to mould the SOM into a topological map, while the rule-construction algorithm was attempting to defy this topology by moving functionally-related neurons together to create a functional map. Consequently, STORM was designed to leave the neurons in their original positions and to represent functional-relationships using lateral connections and the associated temporal-Hebbian learning mechanism.

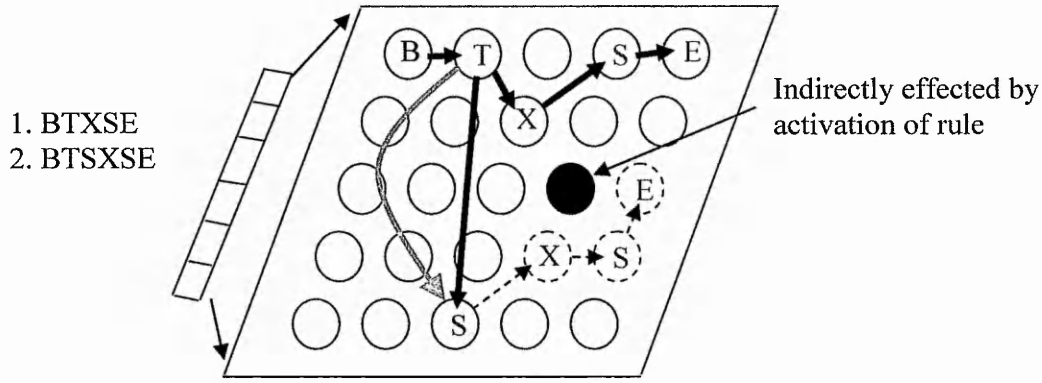


Fig 5.1 – Diagram showing the abrupt context change resulting from the activation of a rule. When the functional-relationship between the 2nd symbol, T, in sequence 1 and the 3rd symbol, S, in sequence 2 is formed, the remaining three symbols in sequence 2 will no longer be activated. This consequently changes the effect of the neighbourhood function, especially on adjacent neurons and causes instability.

A consequence of constructing functional-relationships in this manner, rather than slowly drawing related neurons together, is that the construction of a functional rule is an abrupt event, rather than a slow process. In the conceptual model it was envisaged that by drawing functionally-related neurons together, their contextual representations would slowly change along with the movement of their location. Therefore, subsequent winning neurons in each functionally-related sequence would also change slowly, until eventually they merged into one sequence. However, the effect of abruptly activating a rule is to switch the context of one of the functionally-related neurons to that of its functional-counterpart. The consequence of this is to also

abruptly change the context of the rest of the sequence containing the neuron in question (fig 5.1).

As shown in fig 5.1, when a rule is activated the elements in the sequences following the slave winner (i.e. the last three symbols, *XSE*, in sequence 2) are forced to change position to share the representation used by the master winner. This abrupt change effectively causes a shockwave throughout the map as the influence of the neighbourhood function is changed with respect to potentially every neuron. This effect will be most dramatic for neurons in close proximity to the original winning neurons following the slave winner (i.e. the neuron shown in black in fig 5.1). This effect on the neighbourhood function is compounded by the increased activation of the winning neurons following the master winner. Because the activation of a rule results in the remaining elements in a sequence sharing a single representation, the increased activation of those neurons will also change the effect of the neighbourhood function upon potentially every neuron in the map.

5.1.1 Causality loops

Due to the aforementioned abrupt method of rule-construction, a potential problem arises when the master functional winning neuron (i.e. the winner whose context is used to represent both functionally-related neurons) is itself also a winning neuron preceding the slave winner. As shown in fig 5.2, a causality loop could be created if the winning neuron for the 'X' in sequence 1 attempts to form a slave-like functional relationship with the master neuron representing 'P' in sequence 2. Because the winning neuron for the 'X' in sequence 1 is itself part of sequence 2, changing its context to that of the 'P' neuron will affect every subsequent winning neuron in

sequence 2, including 'P' itself. Therefore, constructing a functional-relationship between the 'P' and the 'X' may prevent the 'P' from being activated. This in turn prevents the functional-relationship between the 'X' and 'P' from being maintained, thus destroying the functional-relationship. In more abstract terms, a causality loop can be defined as "*using an event to change the past, which inadvertently prevents the initial event itself from ever occurring*".

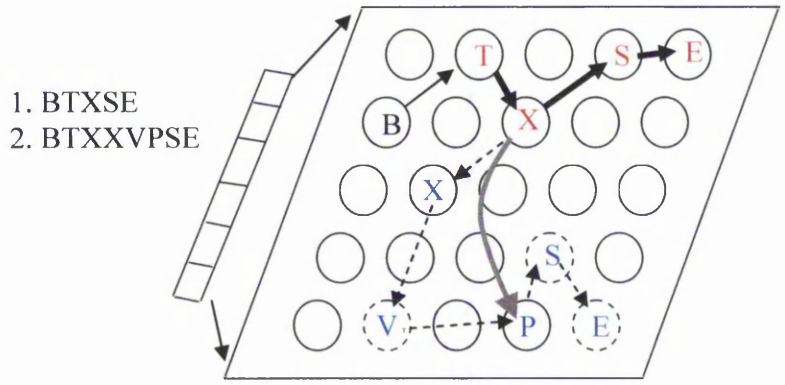


Fig. 5.2 – Diagram showing causality loop resulting from the neuron which represents the X in sequence 1 attempting to form a functional relationship with the neuron representing the P in sequence 2 and thereby changing the latter's context.

While causality loops represent a potential problem for the learning algorithm, the experiments performed on the model have not shown explicit cases where they cause the model to get stuck in a local minima (i.e. a continual cycle of constructing and destroying a specific functional-relationship). The probable reason for this is that if a causality loop does occur, then the instability brought about by destroying the functional-relationship involved acts as interference that disrupts the sequences involved, effectively changing their respective strengths. Because STORM's learning algorithm uses a winner-take-all approach to select the dominant winning neuron (section 4.3.2), the effect of this interference may cause a different dominant winning neuron to be selected next time. Thus, the formation of a causality-loop creates

interference that may prevent it from re-occurring and therefore averting the possibility of infinite causality loops. However, even with the ability to avoid such infinite loops, the interference resulting from a single causality loop may well de-stabilize the model to such a degree that it fails to learn the grammar.

5.2 Recovery from over-generalization

Much of the research into language acquisition has centred on the problem of learning the English past-tense. From the first connectionist PDP models (Rumelhart *et al,* 1986) to the latest symbolic accounts (Pinker, 2000), cognitive researchers have exploited this well documented area of early child language acquisition to attempt to justify their models. The process of learning the English past tense is characterised by a U-shaped learning curve in children's performance (fig 2.5). Following initially good performance on both regular and irregular verbs, children later produce errors involving the inflectional morphology of irregular verbs. These errors involve incorrectly producing the past tense form of an irregular verb by using the suffix from a regular verb (ex. *runned*, instead of *ran*). Following these over-regularisation errors children's performance later improves and they produce the correct forms for past-tense irregular verbs.

Mainstream cognitive researchers (Pinker, 2000; Marcus, 2000) seek to model child language acquisition using rules and symbols. Such an approach interprets the phenomenon of the U-shaped learning curve as evidence that children initially employ rote-memorization to learn both regular and irregular verbs, resulting in an initial steady increase in performance. This is then followed by the acquisition (or activation) of a rule for the production of past tense verbs. While this rule is identified

with respect to regular verbs, it is initially incorrectly applied to all verbs, resulting in the over-regularisation errors seen in irregular verbs. The final phase of the U-shaped learning curve is explained as the formation of exceptions with respect to the past-tense rule. In this phase, the irregular verbs are explicitly re-memorized as exceptions to the rule, resulting in the observed subsequent increase in performance.

The mainstream symbolic account of language acquisition conforms closely to STORM's modus operandi, in that initial memorizations are overridden by rules. However, the final phase of the U-shaped learning curve (i.e. the ability to recover from over-generalization) is not a characteristic of the basic STORM model. While STORM is able to construct rules, once these rules are formed they are applied universally. There is no mechanism to recover from over-generalization. Producing exceptions to rules was unnecessary for a simple language like the Reber grammar. However, more complex languages, even regular grammars, may contain constructs that require exception handling (Figure 5.3).

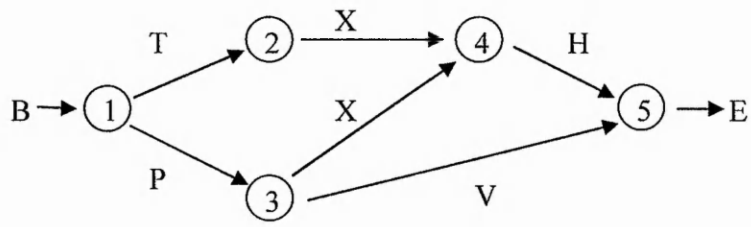


Fig. 5.3 – Simple regular grammar that contains both rules and exceptions.

The simple grammar shown in fig 5.3 illustrates why a memory rule-based mechanism alone is insufficient to learn the grammar. Because the two output symbols labelled '*T*' and '*P*' from state one are functionally-equivalent (i.e. both have the same '*XHE*' ending), STORM could represent them using a rule, with '*T*' as the

master winner and '*P*' as the slave. Thus whenever the '*P*' symbol is encountered in a sequence, the remainder of the sequence would be processed relative to both state two. However, in the grammar state three also has the output symbol '*V*', which is not a valid output symbol for state two. Therefore, if STORM constructs the rule that the '*T*' and '*P*' symbols are functionally-equivalent, it will over-generalize this rule by representing the '*V*' symbol as a valid state two output symbol (figure 5.4). Consequently the model would accept the sequence '*BTVE*', which is not grammatically correct.

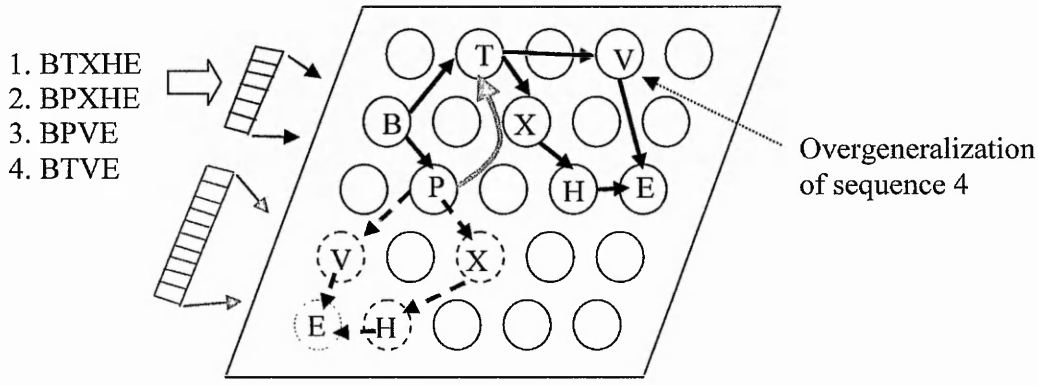


Fig. 5.4 - Illustration of the over-generalization of an exception to a rule. Because of the functional-relationship between the 2nd symbol, T, in sequence 1 and the 2nd symbol, P, in sequence 2, the latter P uses the former T's context for the XHE ending. However, this functional-relationship would also results in the VE ending of in sequence 3 being represented in the context of the 2nd symbol T from sequence 1. Consequently this would allow the model to over-generalize by accepting, as valid, the illegal sequence number 4.

5.2.1 Exception construction mechanism

As stated earlier, mainstream cognitive theory (Pinker, 2000; Marcus, 2000) describes the phenomenon of over-regularization as a U-shaped learning curve. Initial good performance leads to over-regularization errors, which are subsequently corrected as exceptions to the rules are learnt. STORM's learning algorithm currently conforms to

the first two stages of this learning process, namely initial memorization and subsequent generalization. In order to conform to the third stage of this learning process the model must recover from over-generalization using an exception construction-mechanism.

In a memory-rule based model the process of learning exceptions can be perceived as re-memorization, with respect to erroneously applied rules. Because STORM learns rules with respect to a specific common future-context, if a rule is applied to an input whose future-context conflicts with that used to construct the rule, then the rule can be said to have been applied erroneously. For example, in fig 5.4 over-generalization would occur when the 2nd symbol 'T' from sequence 1 forms a functional-relationship with the 2nd symbol 'P' from sequence 2 based on the common future-context 'XHE'. As previously discussed, this would allow the functional-relationship to be applied to sequence three, resulting in over-generalization and acceptance of the erroneous sequence '*BTVE*'.

The rule that functionally-relates the aforementioned symbols would be constructed with respect to the common future-context 'XHE'. Consequently, at the time the functional relationship is created, STORM's back-trace algorithm would be unable to find any existing stored sequence for the 'VE', with respect to the overriding master neuron, when it over-generalizes and applies the rule in the sequence 'BPVE' (figure 5.5). Therefore, without positive evidence to support the application of the rule to the 'VE' ending, STORM would be able to identify this ending as an exception to the rule and then re-memorize it with respect to the winning neuron for the 2nd symbol 'T' from sequence 1 (i.e. re-memorize 'VE' as an exception to the 'XHE' rule).

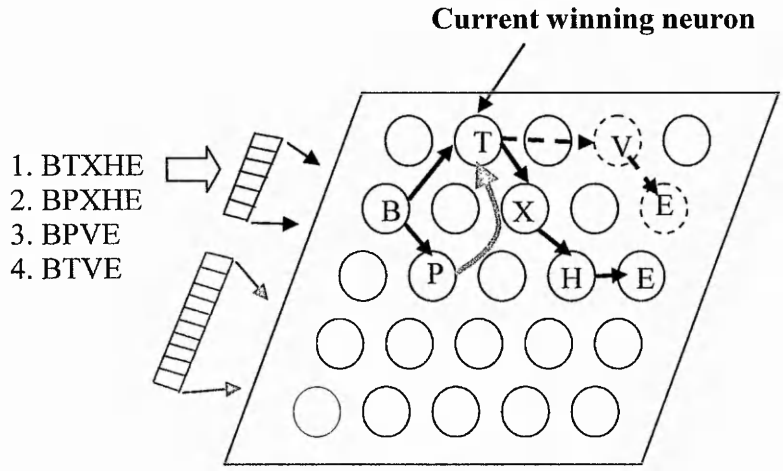


Fig 5.5 – Diagram showing identification of erroneously applied rule. When the 2nd symbol 'P' in sequence 3 is input, the functional-relationship between the 'T' and 'P' neurons (light arrow) is used to override the recurrency mechanism. When the 3rd symbol 'V' from sequence 3 is input, the model uses the 'T' neuron's context. However, the back-trace algorithm is unable to identify a 'VE' ending which corresponds to that 'T' neuron. Therefore, with respect to sequence 4, the 'VE' ending is identified as an exception to the rule involving the T and P neurons.

This process of re-memorizing exceptions to rules can be achieved using the existing temporal-Hebbian learning mechanism combined with a slight change to the rule application algorithm. STORM's current approach to the application of rules is to search for and apply any active rules once the winning neuron is selected and its weights have been updated. However, if the process of rule application is deferred until after the next winning neuron has been selected, then the model can determine whether that next winner is an exception. The algorithm in fig 5.6 describes how this can be achieved by describing the behaviour of the model after the training process is complete. During the training process itself the model would not need to examine lateral exception connections because the learning algorithm has access to the future-context and can therefore perform exception identification.

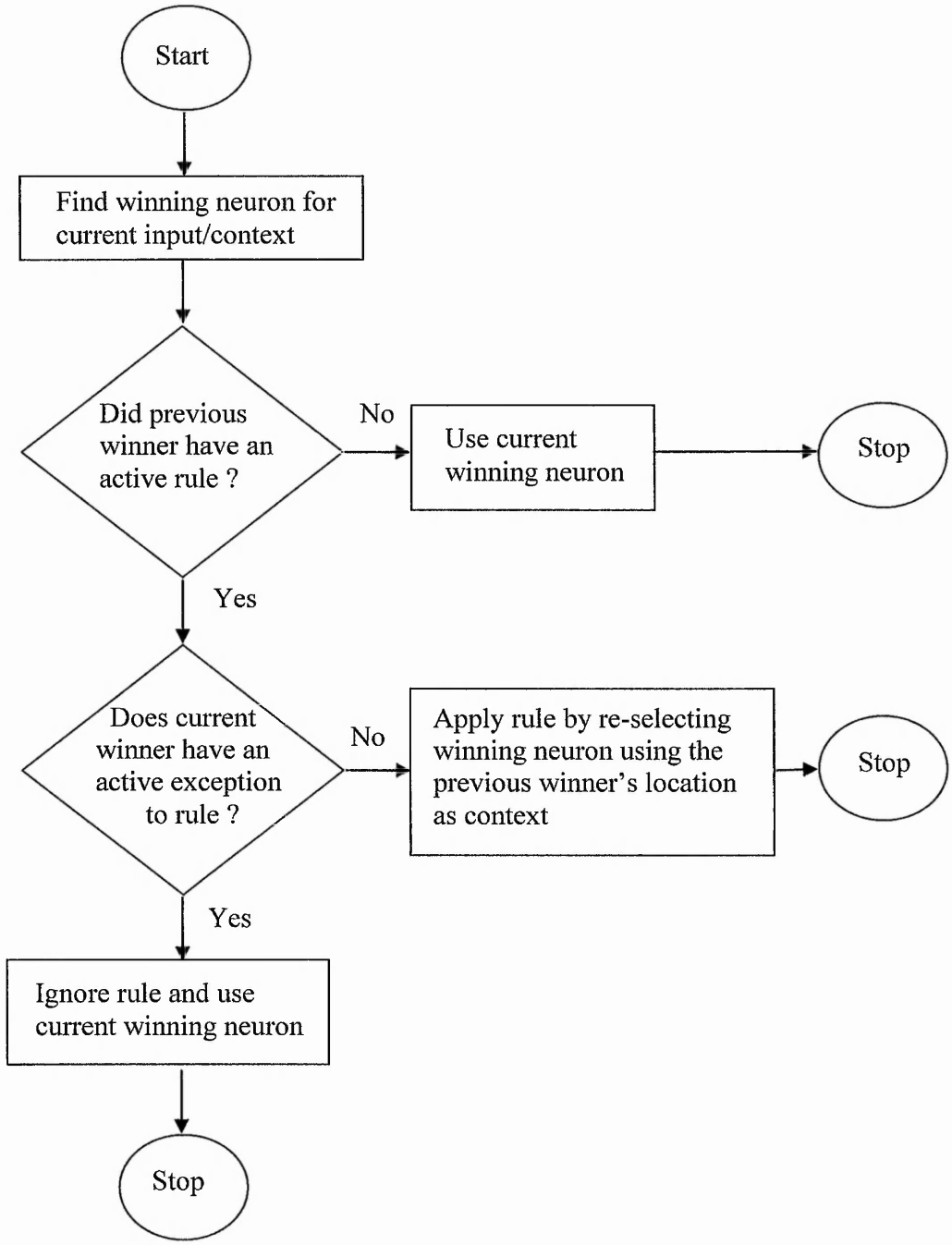


Fig. 5.6 – Revised rule application algorithm to incorporate exception handling mechanism. This algorithm describes the model's behaviour in response to an input pattern after the training process is complete.

By deferring the application of rules until after the next winning neuron has been selected, it becomes possible to identify winning neurons that are specific exceptions

to rules. The construction and identification of exceptions can be achieved using a separate set of exception handling lateral connections. When the learning algorithm detects that a rule is being erroneously applied, it identifies the original winning neuron that would have been selected if the rule didn't exist. By enhancing the lateral exception connections between this original winner and the neuron to which the rule applies, the learning algorithm builds up a relationship between the two neurons. As with the lateral connections representing functional-relationships, once these lateral exception connections exceed a predetermined threshold, the neuron in question can be considered to be an exception to the rule. As detailed in fig 5.6, once a neuron has an active exception to a rule, then the rule application algorithm will ignore the rule and allow the rest of the sequence to be re-memorized as an exception. Fig 5.7 shows a graphical representation of this exception handling mechanism.

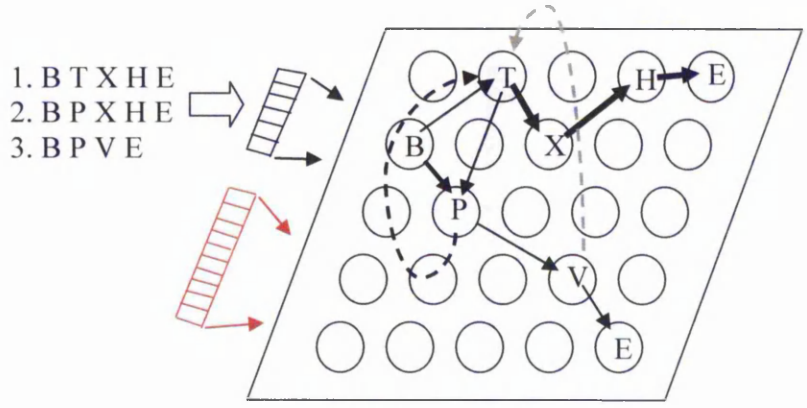


Fig. 5.7 – Diagram showing lateral exception connection (light dotted curved line) between the winning neuron for the V symbol in sequence 3, which represents the exception and the master winning neuron for the T symbol in sequence 1, which represents the rule (dark dotted curved line).

By incorporating an exception handling mechanism, the model now provides an elegant account of how a language learner could recover from over-generalization without needing explicit negative evidence. This approach provides a possible

solution to the age-old logical problem of language acquisition by showing that in a memory-rule based model, recovery from over-generalization can be achieved using a process of re-memorization in response to positive evidence that contradicts an existing rule.

5.3 Context-free grammars

Regular grammars are characterized by allowing only a single terminal symbol on the right hand side of the production rule (i.e. A => x B). This limitation makes regular grammars memory-less and therefore prevents them from describing the type of rules in fig 5.8.

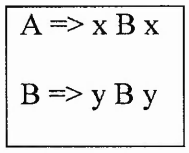


*Fig 5.8 – Example of two rules from a context-free grammar. **x** and **y** are terminal symbols and **A** and **B** are non-terminal symbols.*

Unlike regular grammars, context-free grammars can have any number of terminal and non-terminal symbols on the right hand side of the production rule (see section 2.11 for a discussion of formal linguistic terminology). This enables such grammars to generate rules containing identical non-terminal symbols, preceded and followed by different terminal symbols (fig 5.8). Such grammatical constructs are known as centre-embeddings and constitute a difficult problem for grammar induction models (Weckerly and Elman, 1992).

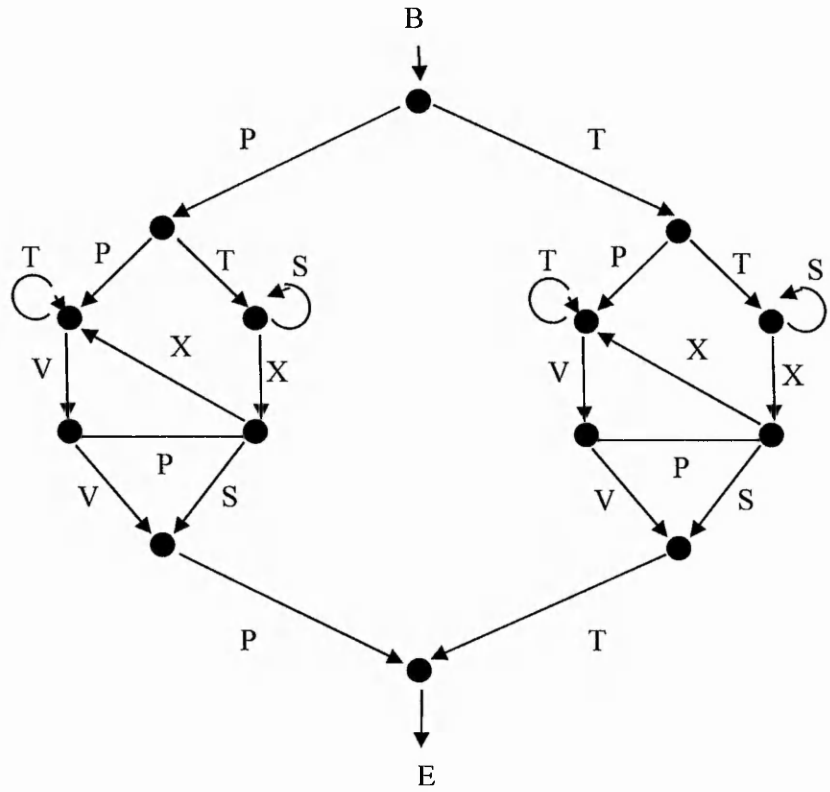


Fig. 5.9 – FSM for Extended Reber grammar.

Models such as the SRN are unable to master languages with deep embeddings because they loose track of information distinguishing the initial terminal symbol (know generally as the problem of long-term dependencies (Bengio and Simard, 1994)). This problem is illustrated in figure 5.9, which shows the extended Reber grammar (Cleeremans, 1989; Fahlman, 1991). In this grammar the second symbol is always the same as the second from last symbol. Therefore in order to correctly predict the second from last symbol, a model must retain information regarding the second symbol in the sequence.

Centre-embeddings also pose a problem for STORM, albeit for different reasons than for conventional DRNs. Because STORM is a memory-rule based model, it must

initially memorize input sequences before it can begin to construct functional-relationships between related neurons. Rules are constructed relative to the context of these original memorizations. However, because rules within the centre-embedding apply independently of the surrounding context, the rule-construction algorithm would need to see examples of the centre-embedding separately in each context (table 5.1). Effectively sequences beginning with the 2nd symbol *P* would be represented completely separately to sequences with a 2nd symbol of *T*.

As shown in table 5.1 STORM will under-generalize centre-embeddings by assuming that the rules of such constructs are specific to the exact context in which they were constructed. Therefore STORM would have to separately learn the grammatical rule for every centre-embedding it appears in. For complex languages this would involve not only redundantly memorizing a lot of sequences, but it would also require training the model on a very large amount of data. Obviously for grammars approaching the complexity of natural language, such a requirement would seriously limit the model's induction capabilities.

#	Extended Reber grammar sequence
1	B T T S X S T E
2	B T T X S T E
3	B P T X S P E
4	B P T S X S P E

Table. 5.1 - Sequences from the extended Reber grammar (figure 5.9) illustrating STORMs inability to learn centre-embeddings. If STORM was trained on sequences 1-3, the model would be able to learn the functional-relationship involving the 4th symbol in sequence 1 (S) and the 3rd symbol in sequence 2 (T). However, it would be unable to generalize this knowledge to the equivalent functional-relationship in sequence 4, despite having been trained on sequence 3.

Essentially the problem of learning centre-embeddings can be perceived as an optimization problem. STORM currently seeks to optimize its internal representations by constructing rules that allow it to use the strongest internal memorization of a sequence. However, this strategy applies only to whole sequences and not to sub-sequences. Because sentences with centre-embeddings effectively contain rules within other rules, STORM must optimize its representation of sub-sequences if it is to efficiently learn such grammars.

An ideal approach would be to use the rule construction mechanism to explicitly re-wire its representation of sub-sequences such that the strongest single sub-sequence would be used, rather than creating redundant representations for every occurrence of a sub-sequence. While such a strategy is very similar to the current approach of using the strongest existing sequence, the construction of sub-sequence rules would require the preservation of context. This requirement is illustrated in figure 5.10, which shows that information concerning the symbol preceding the sub-sequence must be known in order to determine the correct symbol following the sub-sequence.

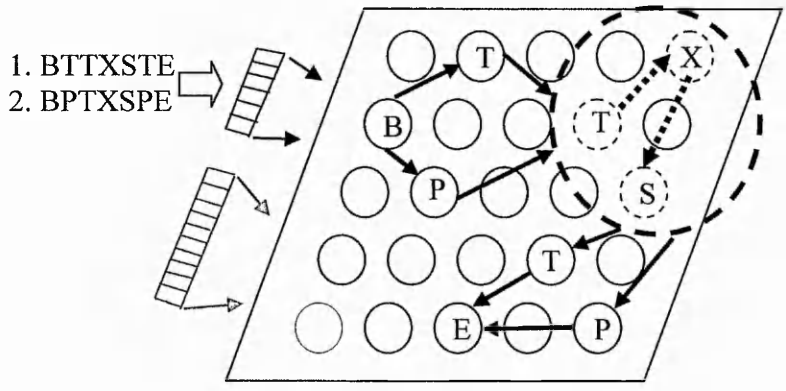


Fig 5.10 – Diagram showing requirement for context preservation when re-using the sub-sequence TXS from the Extended Reber grammar (figure 5.3). The choice of whether a T or a P symbol follows the TXS sub-sequence is determined by the symbol that precedes the sub-sequence.

5.3.1 Sub-sequence rule-construction algorithm

In order for STORM to use an optimized sub-sequence representation to learn context-free grammars, its rule-construction algorithm must be modified. As previously discussed, the current rule-construction algorithm exploits regularities in entire sequences (although it could conceivably just use a finite number of future symbols). Any regularity between stored sub-sequences and input sequences is ignored. Therefore the first stage in such an approach would be to construct a second back-trace algorithm that detects regularities between stored sequences and a specified number of the next symbols in the input sequence (figure 5.11). STORM would then use both the partial and standard back-trace algorithms in conjunction with each other to allow it to detect stored sub-sequences that belong to different sequences than the current input sequence.

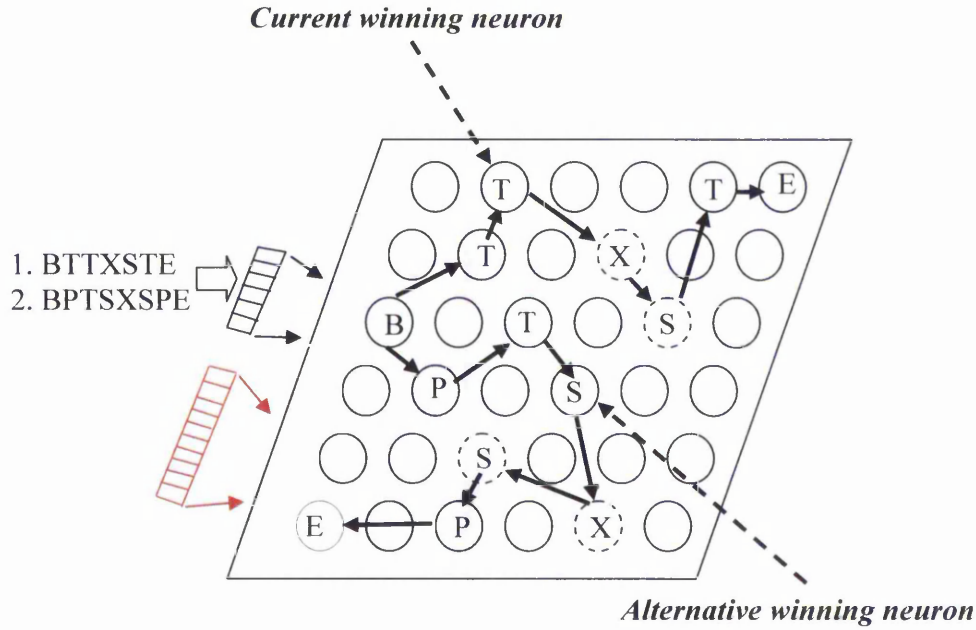


Fig. 5.11 – Diagram illustrating use of partial back-trace algorithm to identify alternative winning neurons in sub-sequences. The alternative winning neuron representing the 4th symbol, S, in sequence 2 has been selected because it is immediate future-context (i.e. the sub-sequence XS) matches that of the winning neuron in sequence 1.

Once a sub-sequence has been identified using the partial back-trace algorithm, STORM would be able to use its existing temporal-Hebbian learning mechanism in almost exactly the same manner as the current model. By building up functional-relationships between the neurons representing symbols preceding the sub-sequence in question, the model would be able re-wire itself to use only a single representation of the sub-sequence. However, as illustrated in figure 5.10, the model's context must then be preserved when processing the sub-sequence. Therefore, the rule-construction mechanism must create a special kind of rule for sub-sequences, which when invoked would push the location of the current winning neuron onto a conventional symbolic stack (Hopcroft and Ullman, 1979). This special rule will also involve popping the representation of the current winning neuron back off the stack after the sub-sequence and incorporating it into the next winning neuron selection algorithm. The position at which to perform these push and pop operations could be identified via the discrepancy between the partial and full back-trace algorithms (i.e. when performing the back traces the push would be performed at the current position, while the pop would occur at the position of the discrepancy between the two back trace algorithms). Thus, this hybrid approach will isolate the sub-sequence (figure 5.10), allowing multiple sequences to utilize the same representation, while still maintaining the distinct context of each sequence.

5.4 Conclusions

The model's sporadic failure to master the Reber grammar may be attributed to a design compromise in the rule-construction algorithm. Due to the use of lateral connections to construct functional-relationships, the activation of a rule is an abrupt event rather than a gradual process. Consequently, the instability this causes may

interfere with the learning process and in some cases may prevent the model from converging. The presence of this instability combined with STORM's stochastic nature may explain why certain models fail to learn a perfect representation of the grammar.

While STORM is currently capable of learning simple regular languages, its learning algorithm needs to be enhanced to allow the induction of more complex grammars. One enhancement that would bring the model closer to natural language would be an exception-construction mechanism. By re-memorizing exceptions to existing rules, STORM would be able to retreat from over-generalizations. This elegant solution to the logical problem of language acquisition would bring STORM's performance closer to that of the U-shaped learning curve characteristic of early human language learners. Context-free languages could also be learnt by optimizing the model's representation of sub-sequences. This would allow it to efficiently learn centre-embeddings without having to see examples of the same grammatical construct in multiple contexts.

6 Conclusions and future work

This research has focused on one of the core problems in artificial intelligence: a problem that has remained elusive, despite almost half a century of investigation by countless researchers. The problem of language acquisition is so daunting that many linguists consider it to be a paradox, a perspective that consequently denies even the possibility of an automated solution. However, given that the primary bottleneck in today's NLP systems is the prerequisite for an incalculable amount of manually derived knowledge, the creation of an automated language acquisition system would constitute a revolutionary breakthrough. STORM represents a significant step towards this breakthrough.

This research has combined the abstract representational power of symbolic rule-based models with the knowledge induction properties of biologically-inspired connectionist models. The result is a model that can learn by example and therefore avoids the prerequisite for manually derived knowledge. This approach uses a representation that encapsulates rules and symbols which allows the model to learn from sparse data. The original aim of the research was to determine whether unsupervised connectionist models could capture the finite-state properties of language. In this respect not only has the project been successful, but it has redefined the original research question by showing that an unsupervised model is capable of exceeding the capabilities of equivalent supervised models, with regard to certain aspects of language acquisition.

In response to the failure of the initial SRSOM to learn the Reber grammar, a revised model, STORM, was designed whose operating principles were derived directly from functional-equivalence theory. Since STORM was designed from its inception to learn a state-based representation, it circumvents the problems that plague other connectionist models when applied to state-based problems. By using regularities between the future-context of stored sequences and that of input sequences, STORM is able to identify functionally-equivalent input symbols. This connectionist memory-rule based approach to grammar induction is both novel and potentially extremely powerful. Identifying functionally-equivalent symbols and binding them together into states in a bottom-up manner, allows the model to learn using a minimum of training data. Such efficiency is a highly desirable characteristic for any model of language acquisition, due to the constraints imposed by the sparse data problem.

Experiments have shown that some STORM models are able to learn a perfect representation of the Reber grammar using a training set of only 30 sequences. A generalization test, using randomly generated sequences that were not encountered during training, shows that the same models can correctly predict the next symbols in the grammar with an accuracy of 100%. An activation analysis of a trained model confirms that STORM forms a state-based representation of the grammar and clusters input symbols based on their respective states. Despite this apparent perfect performance, the experiments also highlight the model's instability by showing that a large number of models fail to learn a perfect representation of the grammar. However, despite this instability STORM's performance is significantly better than for SRNs (Sharkey *et al,* 2000), where only two out of ninety SRNs became finite-state grammar recognizers in a similar experiment on the Reber grammar.

Theoretical extensions to the model show how STORM can be scaled up to learn grammars with a complexity closer to natural language. By treating recovery from over-generalization as re-memorization with a respect to erroneous rules, the model should conform to the third stage in the U-shaped learning curve of child language development and learn exceptions to over-generalized rules. This research also discusses how the model can be extended to learn context-free grammars, by optimizing its representation of sub-sequences.

In conclusion, the research has been successful in creating a novel unsupervised connectionist model, capable of inducting the finite-state properties of a regular grammar. It is therefore a significant step towards automated language acquisition. Furthermore, STORM's foundations in functional-equivalence theory provide a means of generalization that is not available in conventional connectionist models. This powerful learning algorithm may have applications beyond the domain of language acquisition. STORM's approach to grammar induction further challenges traditional nativist perspectives on the feasibility of language acquisition (Chomsky, 1965) by implying that all the information necessary to learn grammar may actually be available in the input sequences themselves.

6.1 Future Research

The proposals for future research will focus on initially stabilizing the model before discussing further experiments on more complex grammatical problems. A number of enhancements to the model will be proposed, along with a general discussion of how STORM could be applied to problems outside of linguistics.

6.1.1 Stabilization of model

Future research will initially focus on stabilizing the model to ensure that STORM is able to learn the grammar more reliably. Since the task of grammar induction involves learning grammars whose structure is potentially unknown, the model must be capable of accurately inducing the grammar to a reliable degree. Otherwise, if an un-reliable model is applied to an unknown problem then any resulting solution will be of questionable value. The reason for this is that without knowledge of the correct grammar, it will be very hard to identify erroneous representations resulting from the model's failure. Therefore, before STORM can be applied to the exploration of unknown grammars, its performance should be improved such that it learns a perfect representation of the grammar over 50% of the time. Once more than 50% of the models can learn the grammar, than averaging techniques can be applied to a set of trained models in order to extract a correct representation of the grammar.

As discussed in section 5.1, a design limitation in STORM's rule-construction algorithm, resulted in a theoretical instability known as a causality loop. This instability arises because the initial activation of a rule is an abrupt event, rather than a gradual process. Consequently the resulting instability may destroy the rule in question. Section 5.1 also discusses how a side-effect of successful rule-construction results in significant instability as well. These instabilities may be responsible for the model's erratic performance. Therefore, re-designing the model to use a smother rule-construction algorithm may significantly improve its performance.

6.1.2 Inflectional morphology, performance evaluation and Mealy machines

As previously discussed, modelling the acquisition of the English past-tense has become the de-facto battleground for competing cognitive learning theories. Connectionist models such as (Rumelhart and McClelland, 1986; Plunkett and Marchman, 1996) attempt to capture the U-shaped learning curve that is believed to characterize the performance of child language learners. However, as discussed by Marcus (2000), no current connectionist model is able to successfully capture the process of recovery from over-regularization, without resorting to implausible manipulation of the training environment or supervisor signal. Therefore, if STORM could be successfully shown to exhibit this U-shaped learning curve during acquisition of the English past-tense, it would establish that the model is a viable connectionist language acquisition system in its own right.

In order to evaluate the model's performance throughout the learning process, its knowledge of the English past-tense must be evaluated to determine whether it conforms to the U-shaped learning curve. Currently the model's performance is evaluated by predicting the two next winning neurons. However, using the simple criterion of selecting the two statistically most probable next winning neuron may not be sufficient to evaluate the model's performance on other grammar. The reason for this is that complex grammars are non-deterministic and may have many possible *next winners.* The probability of each possible winner is determined by its statistical frequency in the training set. One possible method of addressing this issue is to bind additional meaning into the model, such that each input symbol is also associated with an output symbol. This would allow the prediction of the next input symbol which the model would associate with the next specified output symbol.

Using both input symbols and their associated output symbols, STORM would effectively be turned into a Mealy machine. In the context of modelling the acquisition of the English past-tense, the presence of output symbols would dramatically simplify the problem of evaluating the model's performance. This could be achieved by using a representation scheme in which the output symbol denotes the tense of a verb (ex. past, present, future) and the input symbol represents the participle of the verb (ex. –ed, -ing). By applying a simple Hebbian style binding mechanism, the winning neuron representing the current input symbol could be bound to the neuron representing the respective output symbol. Effectively the output symbols would act as semantic tokens that would be bound to the syntactic input symbols. Such an approach would involve training the model by presenting the output symbols denoting the tense of each verb at the same time as the input symbols.

The presence of these semantic output tokens would allow the model to predict the next winning neuron corresponding to a particular meaning (e.g. what would be the past tense participle of a verb, given its infinite form?), rather than just predicting the statistically most likely next winning neuron. In the case of irregular verbs, such participles could consist of a representation of the entire irregular verb (ex. *ran*). Therefore, using this approach the predicted participle for each verb could be evaluated throughout the training process. Thus it would be possible to verify whether STORM conforms to the U-shaped learning curve when learning irregular verbs. Such performance would be confirmed if, as expected, the model exhibits initial rote-learning of irregulars, followed by over-generalization of induced rules (ex. run-ed) and eventual re-memorization of irregulars with respect to erroneously applied rules (section 5.3).

6.1.4 Experiments on more complex grammars

As proposed in section 5.3, STORM could be enhanced to learn context-free grammars. Future research should therefore involve performing the necessary design enhancements that will allow the model to efficiently learn centre-embeddings, possibly involving an external symbolic stack. Once implemented, the model should be initially tested on learning the extended-Reber grammar. If successful, the model could later be applied to grammars with more similarity to natural language, such as those used by Elman (1990). However, as discussed in the previous section, when learning more complex grammars the performance criterion of the next predicted winner may not be sufficient to measure the model's grammar induction abilities. A Mealy machine approach, that links STORM's input symbols to semantic output tokens, may be appropriate for evaluating specific grammatical problems (such as the English past-tense). However, such an approach may not be appropriate for evaluating the model's performance on general grammatical problems. This would be especially true for artificial grammars that have no appropriate meaning which could be attached to the input symbols.

An alternative method of evaluating the performance of these artificial grammars could involve using the criterion of sentence acceptance. By measuring the error level in response to a particular input symbol, the performance algorithm could determine whether the input symbol in question would be deemed acceptable in the grammar. While sentence acceptance is computationally similar to sentence prediction, the latter is more complex because it involves predicting all the possible grammatical symbols at a given state in a sequence. In order fully assess performance, the model must be tested on both positive and negative data (i.e. does the model accept un-grammatical

sequences as well as grammatically correct sequences?). A similar approach to measuring performance has previously been successfully used in experiments (Reali and Christiansen, in press). Sentence acceptance is also common terminology in formal linguistics and would therefore be a self-explanatory choice of performance criterion. However, while sentence acceptance may provide a measure of the model's knowledge of the grammar, it cannot in itself be used to perform many useful tasks (unlike alternative performance criterion such as prediction, which could be used for disambiguation).

6.1.5 Multi-layered STORM model

Section 5.3 detailed an extension to STORM that would allow the model to process context-free grammars. However, while this approach to optimizing sub-sequence representations by using an external stack is technically feasible, it lacks the elegance characteristic of the original STORM model. An alternative approach which could be investigated in future research involves the use of multiple STORM layers operating together (fig 6.1). Section 5.3 explained that STORM was unable to efficiently learn context-free languages due to their centre-embeddings. In a memory-rule based perspective of language, the processing of such constructs can be viewed as a sub-sequence optimization problem.

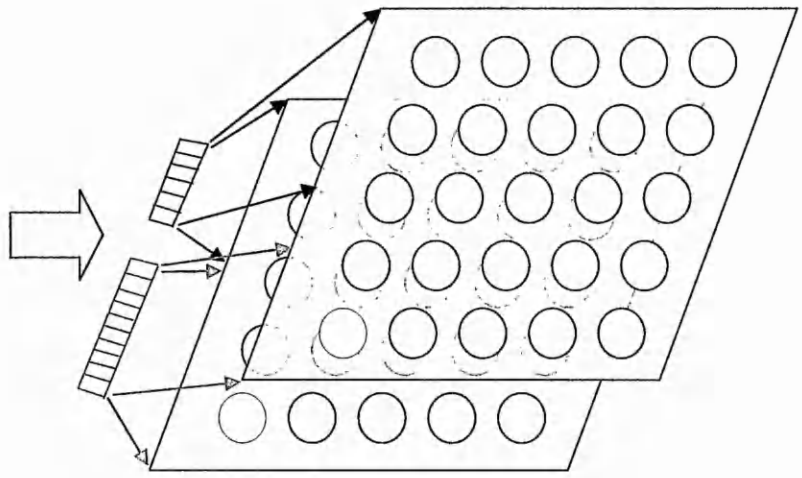


Fig. 6.1 – Multi-layered STORM model

A multi-layered STORM model may be capable of processing context-free languages if the centre constructs (i.e. the sub-sequences within the main sequences) could be handled by the upper layer. This would allow the first layer to learn the regular rules of the grammar, while the second layer learns the rules within the embedded constructs. However, the problem with a multi-layered STORM model concerns the criteria for inter-layer communication. Unlike distributed connectionist models trained via back-propagation, STORM layers cannot just be plugged into each other. A selection criterion in requested to select a particular layer for processing each input symbol presented to the model.

A possible solution to this inter-layer communication problem can be found in the operation of a connectionist system known as neural sequence chunkers (Schmidhuber, 1991). The operational principle of this approach to learning is that the model concentrates only on unexpected inputs, rather than all inputs in the sequence. When an input is not predictable it is passed up to the next layer, an approach which allows the model to find sub-sequences in the inputs. Such a principle of passing

unexpected inputs up to the next layer for processing could be incorporated into a multi-layer STORM model in order to solve the inter-layer communication problem. Thus when an input symbol is presented to STORM, both layers could be searched for the best matching neuron. This should allow sub-sequences (which are more efficient to both represent and predict as separate sequences) to be processed by the upper layer.

6.16 Beyond linguistics – The power of functional generalization

Conventional connectionist models use the past to predict the future. Such models act as similarity engines and learn by generalizing induced knowledge to similar objects/situations as those encountered during training. In contrast, STORM uses the future to understand the structure of the past. STORM learns by generalizing knowledge based on *similarity of behaviour*, rather than just *similarity of form*. Both methods of generalization allow a model to learn by deducing unobserved traits using knowledge of previously induced traits (Pinker, 2000). For example, both conventional connectionist models and STORM would be able to predict that a blackbird is able to fly, based on evidence showing the behaviour of other birds. Conventional connectionist models would solve this problem by generalizing the behaviour of flight to the blackbird because it physically looks like other birds that can fly (i.e. has feathers, wings and a beak). However, STORM would solve this problem by forming a functional-relationship between all of the birds it encounters, based on similarities in their behaviour (i.e. lays eggs, sings at dawn and nests in trees). By learning that all birds are related, STORM would be able to generalize a specific behaviour, such as flight, that it has learnt from other species to the blackbird.

While both conventional connectionist models and STORM can solve problems involving generalization between similar objects, the power of STORM's function based generalizations becomes apparent when learning problems involving dissimilar objects. For example, consider attempting to train a connectionist model to learn that all living organisms will eventually die. Assume such a model was trained on physical descriptions of organisms and abstract descriptions of their physiological behaviour (i.e. reproduces, consumes nutrients, grows). If this model were also provided with training data showing that some of these organisms die, would it be able to generalize that other organisms, encountered during training, also die?

A conventional connectionist model would not be able to adequately solve such a problem because it has no concept of the category *living things*. Therefore, such a model could only deduce that a specific organism would die, if it were physically similar to other organisms that the training data showed would die. However, because STORM learns via function based generalization, it would functionally-relate the organisms encountered during training based on their overlapping physiological activity (effectively forming the category *living things*). Therefore when STORM learns that some of these organisms will die, it is able to generalize that physiological activity to all functionally-related organisms and therefore deduce that all living organisms will die.

This *human-like* ability to learn via function based generalization is potentially very powerful and could be used to model many problems beyond linguistics. Possible connectionist applications include general purpose inductive learning and reasoning

(Heit, 1997), robot planning and control, invariant object recognition (Giles and Maxwell 1987) and financial forecasting.

References

Allen, J. (1995) *NATURAL LANGUAGE UNDERSTANDING, 2nd edition, CA,* The Benjamin/Cummings Publishing Company Inc.

Andersen, P., Hayes, P., Huettner, A., Nirenburg, I., Schmandt, L. and Weinstein, S. (1992) *Automatic Extraction of Facts from Press releases to Generate News Stories: Proceedings of the Third Conference on Applied Natural Language Processing.* Somerset, NJ. p 170-177.

Arnold, D., Balkan, L., Meijer, S., Humphreys, R. and Sadler, L. (1993) *Machine Translation: An Introductory Guide*. Manchester, Blackwell publishers.

Aslin, R., Woodward, J., La Mendola, N. and Bever, T. (1996) Models of word segmentation in fluent speech to infants. *In:* J. Morgan and K. Demuth (eds) *Signal to syntax: bootstrapping from speech to grammar in early acquisition.* Mahwah, NJ, Erlbaum. p. 117-134.

Baker, C. (1979) Syntactic theory and the projection problem. *Linguistic Inquiry*, 10, 533-81.

Barreto, G. and Araújo, A. (2001) Time in self-organizing maps: An overview of models. *International Journal of Computer Research*, 10(2), 139-179.

Bates, E., Thal, D., Trauner, D., Fenson, J., Aram, D., Eisele, J. and Nass, R. (1997) From first words to grammar in children with focal brain injury. *In:* D. Thal and J. Reilly (eds) Special issue on Origins of Communication Disorders, *Developmental Neuropsychology*, 13(3), 275-343.

Beaufays, F., Bourlard, H., Franco, H. and Morgan, N. (2001) NEURAL NETWORKS IN AUTOMATIC SPEECH RECOGNITION. *In: M. Arbib (ed) The Handbook of Brain Theory and Neural Networks, 2nd Edition.* Bradford Books, The MIT Press.

Bengio, Y. and Simard, P. (1994) Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157-166.

Blank, D., Meeden, L. and Marshall, J. (1991) Exploring the symbolic/subsymbolic continuum: A case study of raam. *Technical Report TR332*, Computer Science Department, University of Indiana.

Broeder, Peter and Murre Jaap (eds)(2002) Models of Language Acquisition Inductive and Deductive Approaches, Oxford University Press.

Brown, R. (1973) *A first language: The early stages*. Cambridge, MA, Harvard University Press.

Browne, A. and Sun, R. (2000) Connectionist variable binding. *In:* S. Wermter and R. Sun (eds) *Hybrid Neural Systems*. Heidelberg, Springer Verlag.

Cairns, P., Shillcock, R., Chater, N. and Lew, J. (1997) Bootstrapping word boundaries: a bottom-up corpus based approach to speech segmentation. *Cognitive Psychology*, 33, 111-153.

Chalmers, D. (1990) Syntactic transformations on distributed representations. *Connection Science*, 2(1-2), 53-62.

Chappel, G. and Taylor, J (1993) The temporal Kohonen map. *Neural Networks*, 6, 441-445.

Chomsky, N. (1959) On certain formal properties of grammars. *Information and control 2*, 2, 137-167.

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

Chomsky, N. (1972) *Language and Mind*, 2nd edition. New York, Harcourt, Brace and World.

Chomsky, N (1981) *Lectures on Government and Binding*. Foris Publications, Dordrecht.

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

Cohen, J and Servan-Schreiber, D. (1992) Context, cortex and dopamine: A connectionist approach to behaviour and biology in Schizophrenia. *Psychological Review*, 99, 45-77.

Cohen, N. and Squire, L. (1980) Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210, 207-210.

Corrigan, R. and Iverson, G. (eds) (1994) *The reality of linguistic rules.* Amsterdam, Benjamins.

Craven, M. and Shavlik, J. (1997) Using Neural Networks for Data Mining. *Future Generatation Computer Systems: special issue on data mining*, 13(2-3), 211-229.

Damasio, A. and Damasio, H. (1992) Brain and Language. *Scientific American*, 267(3), 89-95.

Davis, M. (2003) Connectionist modelling of lexical segmentation and vocabulary acquisition. *In:* P. Quinlan (ed) *Connectionist models of development: Developmental processes in real artificial neural networks*. Psychology Press, Hove, UK.

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

Elman, J., Bates, E., Johnson, M., Karmiloff-Smith, A., Parisi, D. and Plunkett, K. (2001) *Rethinking Innateness: A connectionist Perspective on Development*. Cambridge, MA. MIT Press.

Euliano, N. and Principe, J. (1996) A Self-Organizing Temporal Pattern Recognizer with Application to Robot Landmark Recognition. *In:* Sintra Spatiotemporal Models in Biological and Artificial Systems Workshop.

Fahlman, S. (1991) The recurrent cascade-correlation architecture. *In:* R. Lippmann *et al*. (eds) *Advances in Neural Information Processing Systems 3*, San Mateo, CA, Morgan Kaufmann. p. 190-196.

Farah, M. and McClelland, J. (1991) A computational model of semantic memory impairment: modality specificity and emergent category specificity. *Journal of Experimental Psychology: General*, 120, 339-357.

Fitch, T. and Hauser, M. (2004) Computational Constraints on Syntactic Processing in a Nonhuman Primate. *Science,* 303, 377-380.

Fodor, J. (1983) *The Modularity of Mind.* Cambridge, MA. MIT Press.

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

French, R. (1999) Catastrophic forgetting in connectionist networks. *Trends in Cognitive Science*, 3(4), 128-135.

Giles, C. and Maxwell, T. (1987) Learning, invariance and generalization in high-order neural networks. *Applied Optics,* 26(23), p 4972-4978.

Gold, E. (1967) Language Identification in the Limit. *Information and Control*, 16, 447-474.

Gordon, P. (2004) Numerical Cognition Without Words: Evidence from Amazonia. *Science*. 306. p. 496-499.

Gray, F. (1953) Pulse Code Communication. *U.S. Patent 2 632 058.*

Grossberg, S. (1976) Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors, *Biological Cybernetics*, 23, 121-134.

Hadley, R. (1994) Systematicity in connectionist language learning. *Mind and Langauge*, 9(3), 247-272.

Hadley, R and Cardei, V (1999) Language acquisition from sparse input without error feedback. *Neural Networks*, 12(2), 217-235.

Hebb, D. (1949) *The organization of behaviour*. New York, Whiley.

Heit, E. (1997) Features of similarity and category-based induction. *In: Proceedings of the Interdisciplinary Workshop on Similarity and Categorisation.* Edinburgh. p. 115-121.

Hinton, G. and Shallice, T. (1991) Lesioning an attractor network: Investigation of acquired dyslexia. *Psychological Review*, 98, 74-95.

Honkela, T., Pulkki, V. and Kohonen, T. (1995) Contextual relations of words in Grimm tales analyzed by self-organizing map. In: *Proceedings of International Conference on Artificial Neural Networks, ICANN-95,* volume 2, Paris, France. p. 3-7.

Hopgood, A. (2001) Intelligence Systems for Engineers and Scientists, 2nd edition, Florida, LLC Press.

Hopcroft, J. and Ullman J. (1979) INTRODUCTION TO AUTOMATA THEORY, *LANGUAGES AND COMPUTATION.* Addison-Wesley.

Horning, J. (1969) *A study of grammatical inference.* PhD thesis, Stanford University, California.

Humboldt, W. (1836/1972) *Linguistic variability and intellectual development.* G. Buck and F. Raven (eds). Trans. Philadelphia: University of Pennsylvania Press.

Jackendoff, R. (2002) *FOUNDATIONS OF LANGUAGE: Brain, Meaning, Grammar, Evolution.* Oxford University Press.

Jackson, D., Constandse, R. and Cottrell, G. (1996) Selective attention in the acquisition of the past tense. *In: Proceedings of the 18th Annual Conference of the Cognitive Science Society.* Hillsdale, NJ. p. 183-188.

James, D and Miikkulainen, R. (1995) SARDNET: A self-organizing feature map for sequences. *In:* G. Tesauro *et al.* (eds) *Advances in Neural Information Processing Systems*, volume 7, Cambridge, MIT Press, p. 577-584.

Kangas, J. (1990) Time-Delayed Self-Organizing Maps. In: *Proceedings of the International Joint Conference on Neural Networks*, Volume 2. p 331-336.

Kasslin, M., Kangas, J. and Simula, O. (1992) Process State Monitoring Using Self-Organizing Maps. *In:* I. Aleksander and J. Taylor (eds) *Artificial Neural Networks*, volume 2, Amsterdam, Netherlands. p. 1531-1534.

Kohonen, T. (1991) The Hypermap Architecture. *In:* T. Kohonen *et al.* (eds) *Artificial Neural Networks*, North-Holland. p. 1357-1360.

Kohonen, T. (2001) *Self-Organizing Maps*, 3rd edition, Germany, Springer-Verlag.

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

Kolen, J. (1994) Fool's gold: Extracting finite state machines from recurrent network dynamics. *In:* J. Cowan *et al.* (eds) *Advances in Neural Information Processing Systems*, volume 6, Morgan Kaufmann, p. 501-508.

Lagus, K., Honkela, T., Kaski, S. and Kohonen, T. (1999) WEBSOM for textual data mining. *Artificial Intelligence Review*,13, p 345-364.

Lane, P. and Henderson, J. (1998) Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding. *In: Proceedings of the International Conference on Artificial Neural Networks*, Skovde, Sweden. p. 615-620.

MacWhinney, B (2002a) The gradual emergence of language. *In*: GivÓn, T. and Malle, B. (Eds). *The evolution of language out of pre-language*. Amsterdam, John Benjamins. p 231-263.

MacWhinney, B. (2002b) Lexical Connectionism. *In:* P. Broeder and J. Murre (eds) *Models of Language Acquisition: Inductive and Deductive Approaches*. Oxford University Press. p 9-28.

MacWhinney, B (2004) A multiple process solution to the logical problem of language acquisition. *Journal of Child Language*, 31, 883-914.

Marcus, G. (1998) Rethinking Eliminative Connectionism. *Cognitive Psychology*, 37, 243-282.

Marcus, G. (2000) Children's Overregularization and Its Implications for Cognition. *In:* P. Broeder and J. Murre (eds) *Models of Language Acquisition: Inductive and deductive approaches.* Oxford, Oxford University Press. p. 154-176

Mayberry III, M. and Miikkulainen, R. (1999) SARDSRN: A neural network shift-reduce parser. *In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden.

McCulloch, W. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115-133.

McLeod, P., Plunkett, K. and Rolls, E. (1998) *Introduction to Connectionist Modelling of Cognitive Processes*. Oxford University Press.

Minsky, M. and Papert, S. (1969) *Perceptrons*. Cambridge, Massachusetts, The MIT Press.

Mozer, M. (1993) Neural net architectures for temporal sequence processing. *In:* A. Weigend and N. Gershenfeld (eds) *Time Series Prediction: Forecasting the future and understanding the past*, Addison Wesley, p 243-264.

Omlin, C. (2001) Understanding and Explaining DRN Behaviour. *In:* J. Kolen and S. Kremer (eds) *A Field Guide to Dynamic Recurrent Networks*. New York, IEEE Press. p. 207-227.

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

Pinker, S. (2000) *Words and Rules: The Ingredients of Language*. London, Phoenix.

Platt, R., Brock, O., Fagg, A. and Karupiah, D., Rosenstein, M., Coelho, J., Huber, M., Piater, J., Wheeler, D and Grupen, R. (2003) *A Framework For Humanoid Control and Intelligence: Proceedings of the 2003 IEEE International Conference on Humanoid Robots*. Karlsruhe Munich, Germany.

Plunkett, K., Sinha, C., Møller, M. and Strandsby, O. (1992) Symbol grounding or the emergence of symbols? Vocabulary growth in children and a connectionist net. *Connection Science*, 4, 293-312.

Plunkett, K. and Marchman, V. (1996) Learning from a connectionist model of the English past tense. *Cognition*, 61, 299-308.

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

Pollard, C. and Sag, I. (1994) *Head-driven Phrase Structure Grammar*. The University of Chicago Press.

Reali, F. and Christiansen, M. (in press) Uncovering the richness of the stimulus: Structure dependence and indirect statistical evidence. *Cognitive Science*.

Reber, A. (1967) Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior,* 5, 855-863.

Ritter, H. and Kohonen, T. (1989) Self-Organizing semantic maps. *Biological Cybernetics, 61, 241-254.*

Ritter, H., Martinetz, T., Schulten, K. (1992) *Neural Computational and Self-Organizing Maps*. MA, Addison-Wesley, Reading.

Roe, A., Pallas, S., Kwon, Y. and Sur, M. (1992) Visual Projections Routed to the Auditory Pathway in Ferrets: Receptive Fields of Visual Neurons in Primary Auditory Cortex. *The Journal of Neuroscience*, 12(9), 3651-3664.

Rumelhart, D., Hinton, G. and Williams R. (1986) Learning internal representations by back-propagating errors. *Nature*, 323, 533-536.

Rumelhart, D and McClelland, J. (1986) On learning the past tense of English verbs. *In:* J. McClelland and D. Rumelhart (eds) *Parallel distributed processing*, Vol. 2. MIT Press, Cambridge, MA. p. 216-271.

Sapir, E. (1929) The Status of Linguistics as a Science. *In:* E. Sapir (1958) *Culture, Language and Personality*. Berkeley, CA. University of California Press.

Schmidhuber, J. (1991) Neural Sequence Chunkers. *Technical Report FKI-148-91.* Technische Universität München, Germany.

Sejnowski, T. (1986) Open questions about computation in the cerebral cortex. *In:* J. McClelland and D. Rumelhart (eds) *Parallel distributed processing*, Volume 2, Cambridge, MA. p. 372-389.

Sharkey, N., Sharkey, A. and Jackson, S. (2000) Are SRN's sufficient for modelling language acquisition? *In:* P. Broeder and J. Murre, *Models of Language Acquisition: Inductive and Deductive Approaches*. Oxford University Press. p. 33-54.

Stump, G (2001) *Inflectional Morphology: A Theory of Paradigm Structure.* Cambridge University Press.

Varsta, M., Milan, J. and Heikkonen, J. (1997) A recurrent self-organizing map for temporal sequence processing. *In: Proceedings of the ICANN'97: International Conference on Artificial Neural Networks.* Springer-Verlag. p. 421-426.

Venter, J. *et al.* (2001) The Sequence of the Human Genome. *Science*. 291(5507), 1304-1351.

Voegtlin, T. and Dominey, P. (2002) Recursive self-organizing maps. *Neural Networks*, 15(8-9), 979-991.

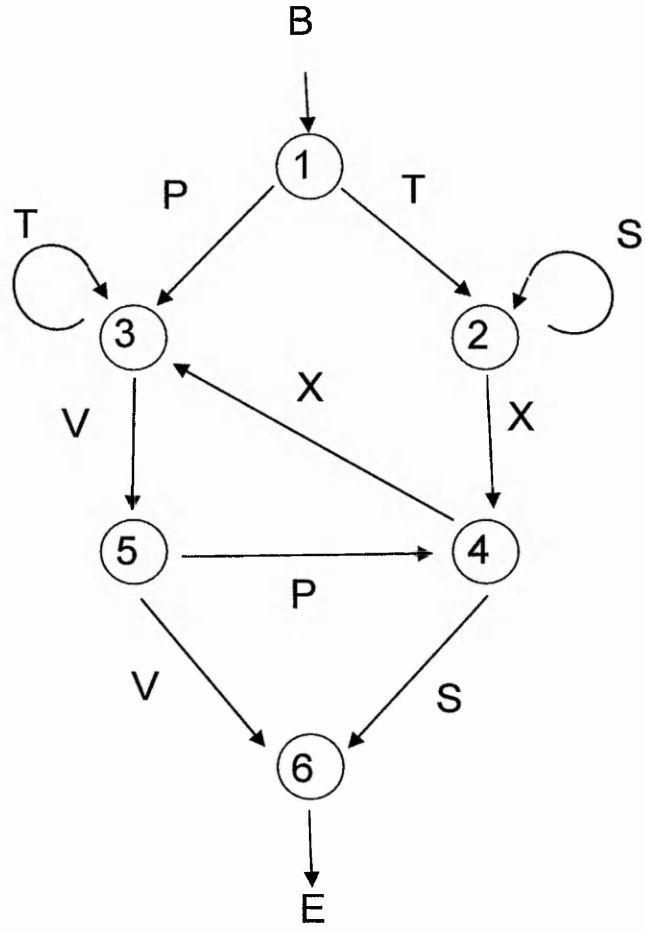
Webster, M., Bachevalier, J. and Ungerleider, L. (1995) Development and plasticity of visual memory circuits. *In:* B. Julesz and I. Kovacs (eds) *Maturational Windows and Adult Cortical Plasticity.* Sante Fe Institute Studies in the Science of Complexity, Proceedings Vol. XXIII. Reading, MA. Addison-Wesley. p. 73-92.

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

Yao, J and Tan, C. (2001) *Guidelines for Financial Forecasting with Neural Networks: Proceedings of International Conference on Neural Information Processing.* Shanghai, China. p 772-777.

Zeng, Z., Goodman, R. and Smyth, P. (1993) Learning finite state machines with self-clustering recurrent networks. *Neural Computation*, 5(6), 976-990.

Appendix A - Finite-state machine for Reber grammar



Appendix B – Glossary

Acronym	Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
ART	Adaptive Resonance Theory
CFG	Context-Free Grammar
CSG	Context-Sensitive Grammar
DFA	Deterministic Finite-Automata
DNA	Deoxyribose Nucleic Acid
DRN	Dynamic Recurrent Network
DVD	Digital Versatile Disc
FSG	Finite-State Grammar
FSM	Finite-State Machine
JASPER	Journalist's Assistant for Preparing Earning Reports
LIN	Leaky Integrator Neuron
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
PDP	Parallel Distributed Processing
RAAM	Recursive Auto-Associative Memory
RSOM	Recurrent Self-Organizing Map
SARDNET	Sequential Activation Retention and Decay NETwork
SRN	Simple Recurrent Network
SOM	Self-Organizing Map
SOTPAR	Self-Organizing Temporal Pattern Recognizer

SRSOM	Simple Recurrent Self-Organizing Map
SSN	Simple Synchrony Network
STORM	Spatio-Temporal Self-Organizing Recurrent Map
TKM	Temporal Kohonen Map
TSOM	Temporal Self-Organizing Map
TSP	Temporal Sequence Processing
TSVB	Temporal Synchrony Variable Binding
XOR	Exclusive OR

Appendix C – Published works

McQueen, T., Hopgood, A., Tepper, J. and Allen, T. (2002) A Recurrent Self-Organizing Map for Temporal Sequence Processing. *In: Proceedings of the 4th International Conference in Recent Advances in Soft Computing (RASC2002).* UK, The Nottingham Trent University.

McQueen, T., Hopgood, A., Allen, T. and Tepper, J. (2004) Extracting Finite Structure from Infinite Language. *In:* Bramer, M. *et al* (eds) *Research and Development in Intelligent Systems XXI: Proceedings of AI-2004, the Twenty-fourth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence.* Springer.

McQueen, T., Hopgood, A., Allen, T. and Tepper, J. (in press) Extracting Finite Structure from Infinite Language. *Knowledge Based Systems.*

A Recurrent Self-Organizing Map for Temporal Sequence Processing

T. A. McQueen, A. A. Hopgood, J. A. Tepper and T. J. Allen
Department of Computing & Mathematics,
The Nottingham Trent University
Burton Street, Nottingham, NG1 4BU, United Kingdom
e-mail: {thomas.mcqueen, adrian.hopgood, jonathan.tepper, tony.allen}@ntu.ac.uk

Abstract: *We present a novel approach to unsupervised temporal sequence processing in the form of an unsupervised, recurrent neural network based on a self-organizing map (SOM). A standard SOM clusters each input vector irrespective of context, whereas the recurrent SOM presented here clusters each input based on an input vector and a context vector. The latter acts as a recurrent conduit feeding back a 2-D representation of the previous winning neuron. This recurrency allows the network to operate on temporal sequence processing tasks. The network has been applied to the difficult natural language processing problem of position variant recognition, e.g. recognising a noun phrase regardless of its position within a sentence.*

Keywords: *neural network, natural language processing, temporal sequence processing, self-organizing map, unsupervised, recurrent.*

1. Introduction

Temporal sequence processing (TSP) is an increasingly important field for neural networks, with applications ranging from weather forecasting to speech recognition [1]. TSP involves the processing of signals that vary over time. Problems such as predicting the weather generally cannot be solved by just examining a set of current inputs from the dynamic system in question, e.g. a satellite image showing today's cloud cover. Rather, any prediction must be based on the current input in the context of a number of previous inputs, e.g. a satellite image for today along with satellite images from the previous five days, showing how the weather has changed so far over the week.

Neural network models for TSP outperform alternative methods, such as NARMAX [9], mainly due to their ability to learn and generalise when operating on large amounts of data [9]. Supervised learning is usually used to solve TSP problems, i.e. the recurrent neural network must be explicitly trained by providing a desired target signal for each training exemplar. Current supervised learning methods are computationally inefficient [8] and are unable to solve certain types of problems [6].

A number of unsupervised neural networks for TSP have been proposed [6], mostly based on the self-organizing map (SOM) [5]. These models use a variety of external and internal memory mechanisms to capture information concerning past inputs, e.g. tapped delay lines and leaky integrators. Unsupervised learning has advantages over equivalent supervised techniques in that it makes fewer assumptions about the data it processes, being driven solely by the principles of self-organization, as opposed to an external target signal.

We present a novel, unsupervised, recurrent neural network based on a SOM to identify temporal sequences that occur in natural language, such as syntactic groupings. The network uses both an input vector and a context vector, the latter of which provides a 2-D representation of the previous winning neuron. The proposed network is applied to the difficult natural language processing (NLP) problem of position variant recognition, e.g. recognising a noun phrase regardless of its position within a sentence.

2. Architecture and algorithm

The network has a 28-bit input vector that provides a binary representation of the input tag being processed. In addition to this input vector, the network also uses a second context vector. The size of this context vector can be varied depending on the size of the network, but in experiments detailed below the context vector was set to 10 bits (Fig. 1). Both the input and the context vector are used in the Euclidean distance calculation to determine the winning neuron in a similar manner to a standard SOM.

The context vector represents the previous winning neuron using a 10-bit coordinate vector. The first five bits of this vector represent the binary number of the winning neuron's row, while the latter five bits represent the binary number of the winning neuron's column. This is an efficient method of coordinate representation that provides the network with a 2-D view of spatial context. It is an improvement over an initial approach, which represented the previous winning neuron using only a binary representation of its number within the SOM. Such a representation prevented the network from seeing similarities between neighbouring neurons in adjacent columns. For example, neuron 8 and neuron 28 are neighbours on the SOM shown above and will therefore be representative of similar patterns. However, the binary representation of the numbers 8 (i.e. 01000) and 28 (i.e. 11100) are dissimilar. Thus similar input patterns may result in dissimilar context causing similar sequences to be clustered to significantly different regions of the SOM. It is envisaged that this would reduce the network's ability to generalize.

The coordinate system of context representation solves this problem by effectively providing the network with a 2-D view of winning neurons. In the example given above, neuron 8 would be represented as 1000000010, while neuron 28 would be represented as 0100000010. (Note that only one bit is different in this example as opposed to two bits in the example above).

As with the standard SOM, the recurrent SOM presented here uses a neighbourhood function to update the weights of neurons in a region around the winning neuron. Both the weight vector and the context vector of neighbouring neurons are moved towards those of the respective input and context vectors. The network uses a Gaussian neighbourhood function to calculate the learning rate that will be applied to these neurons. This function allows immediately neighbouring neurons to experience similar weight changes to those of the winning neuron, while distant neurons experience minimal weight changes. However, in order to improve computational efficiency, the neighbourhood function uses a cut-off value, beyond which neurons do not take part in weight updates at all.

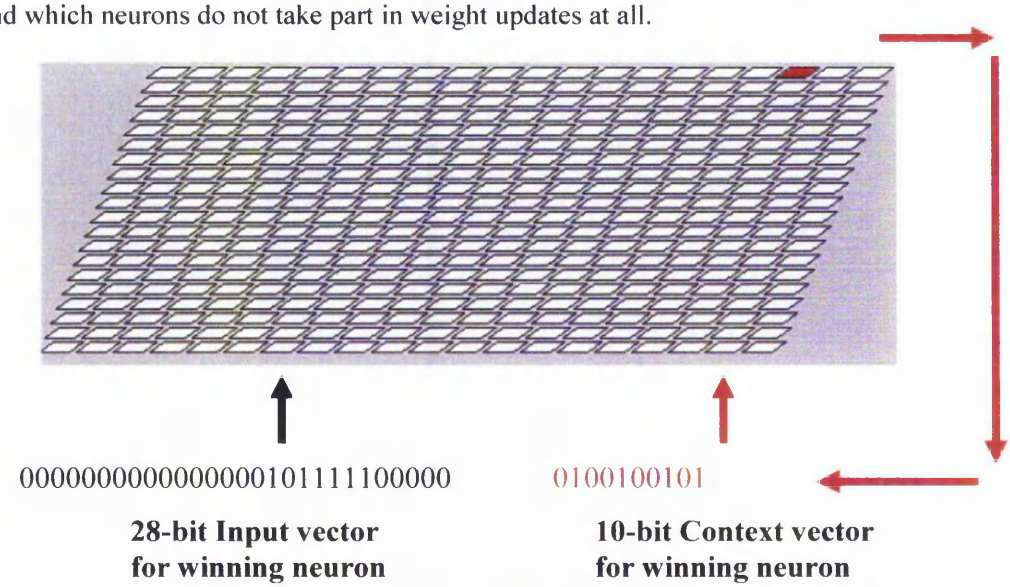


Fig. 1 – Network showing recurrent feedback

3. Experiments

Initially, the new network is being applied to a corpus-based natural language task (Fig. 2) using the Lancaster Parsed Corpus (LPC) [7]. At present, the main objective of the research is to identify coarse phrase boundaries (e.g. noun phrases or verb phrases with little or no embedding) that may emerge on the topological map from exposure to linear sequences of words (sentences) that have been pre-tagged with symbols denoting the word's part-of-speech (e.g. noun, adjective, verb etc) [2].

A network with an output layer of 20 × 20 neurons was trained in two phases, following Kohonen's research on training SOMs [3]. The first *convergence* phase consisted of 1000 epochs, in which the learning rate was linearly reduced from an initial value of 0.1, but was not allowed to fall below 0.01. This was followed by a second *fine-tuning* phase in which a learning rate of 0.01 was applied for 2500 epochs. While the number of epochs in the first phase conforms with Kohonen's research [3], the number of epochs in phase two is considerably smaller than the number suggested. At this initial stage in the research, this reduction is necessary due to time and computational constraints. However, experimental analysis has not shown a significant reduction in the quality of results when training times in phase two are reduced.

A sample of 664 sentences from the LPC [7] were presented to the network. Presentation occurred in random order to improve training efficiency and to prevent the weights from becoming stuck during the low neighbourhood value in phase two. The context vector is set to zero between each sentence to prevent contextual information from previous sentences interfering with subsequent sentences.

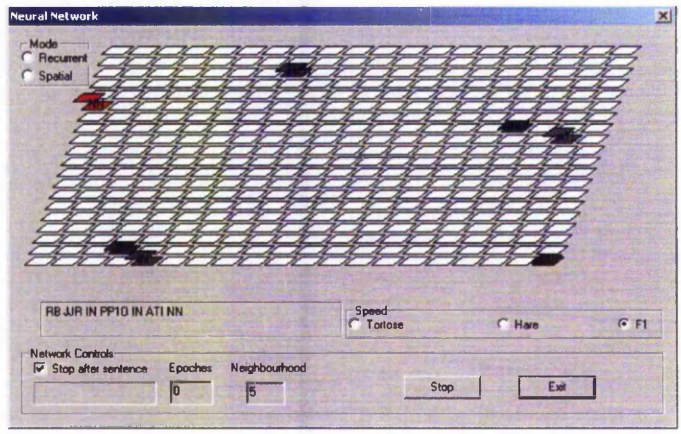


Fig. 2 – Screenshot from the current network. The raised, coloured polygons represent winning neurons for the sentence of tags presented to the network.

4. Results

The preliminary results are encouraging, as they show that word tags are being clustered in locations consistent with their context. The results in Figs. 3–5 show three simple artificially constructed sentences of varying tense. Despite these variations in tense, each exhibits a similar trace pattern over the map. We refer to these traces as *signatures*.

Fig. 6 shows two simple noun phrases with and without a preposition. While both sentences show similar signatures for the noun phrase, the effect of the preposition can clearly be seen to alter the signature of the second phrase.

It is hoped that further analysis will reveal the extent to which the network can exploit the context and show what kind of temporal syntactic patterns the network can find in input

sequences. A major benefit of finding such patterns in an unsupervised manner is that, unlike supervised techniques, there is no dependency on manually annotated corpora, which are not widely available due to the high costs associated with manually annotating raw language data. In fact it is envisaged that, should the unsupervised system prove successful in extracting syntactic structure, it would serve as an automatic syntactic annotation system thus reducing the need and cost of manual annotation.

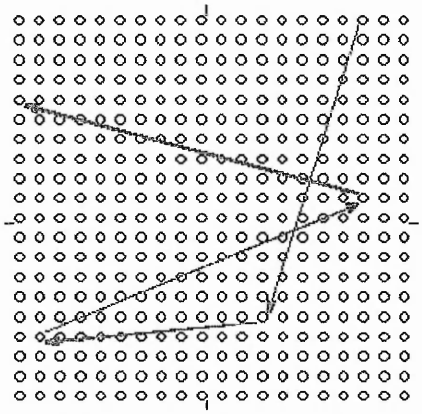


Fig. 3 – Signature for sentence: "*she goes down the stairs*"

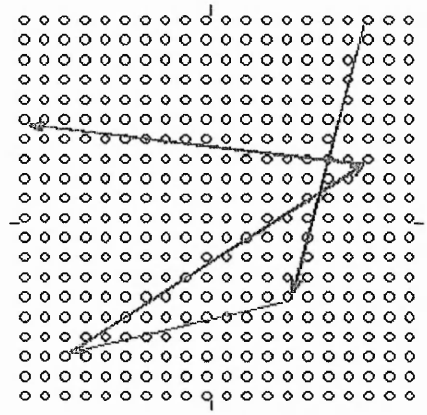


Fig. 4 – Signature for sentence: "*she went down the stairs*"

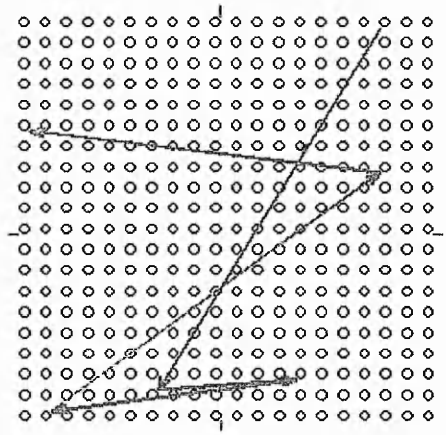


Fig. 5 – Signature for sentence: "*she is going down the stairs*"

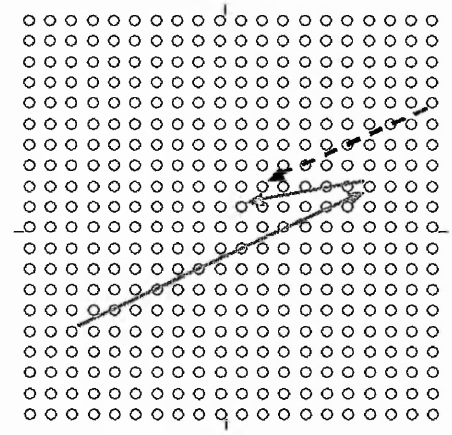


Fig. 6 – Noun phrase with and without preposition

The home

In the home

5. Conclusions and future work

We have presented a novel recurrent SOM and applied it to the problem of position-variant recognition. We have shown that the network forms signatures in response to temporal sequences present in the inputs.
In addition to the natural language task, research is also being conducted into enhancing the recurrent SOM using lateral connections and a temporal Hebbian learning [4] mechanism. The purpose of such a mechanism is to attempt to control the recurrency, allowing feedback to occur only when the winning neurons, whose representations are to be fed-back, are stable. This temporal Hebbian learning mechanism has been used in a previous experimental neural network and it is hoped that it will reduce the SOM's training time.
In the next phase of this investigation, hierarchical clustering methods based on temporal SOMs will be developed to obtain finer-grained syntactic groupings. Future work will focus on the context representation that is fed back. The representation may be enlarged to give more emphasis to the context vector than the input vector, and it may also be optimised using genetic algorithms. Further experiments will be performed in the domain of natural language processing; specifically the network will be used to attempt to detect phrase boundaries. Additionally, if the network proves successful, it may also be used in a number of other areas including computer virus detection, speech recognition and image analysis.
On a wider scale, the recurrent SOM could be used as the core of a temporal neural processing system. For example, the recurrent SOM clusters patterns based on input featural similarities whilst a supervised neural network uses these reduced representations to perform a mapping to a corresponding set of desired outputs.

6. References

[1] G. A. Barreto and A. F .R. Arajo (2001). Time in self-organizing maps: An overview of models. *International Journal of Computer Research: Special Issue on Neural Networks: Past, Present and Future*, 10(2):139-179.

[2] R. Garside, G. Leech and T. Varadi (1987). Manual of information to accompany the Lancaster Parsed Corpus. Department of English, University of Oslo.

[3] S. Haykin (1999). Neural Networks: A Comprehensive Foundation, Prentice Hall

[4] D. Hebb (1949). The Organization of behaviour, John Wiley.

[5] T. Kohonen (1984). Self-Organization and Associative Memory, Springer-Verlag.

[6] M. C. Mozer (1994). Neural net architectures for temporal sequence processing, in A. S. Weigend and N. A. Gershenfeld (eds), *Time Series Prediction: Predicting the Future and Understanding the Past*, pp 243–264, Addison-Wesley.

[7] J. A. Tepper, H. M. Powell and D. Palmer-Brown (2002). A corpus-based connectionist architecture for large-scale natural language parsing. *Connection Science*, 14 (2).

[8] J. Schmidhuber (1991). Adaptive history compression for learning to divide and conquer, *Int. Joint Conf. on Neural Networks*, Vol 2, pp 1130–1135.

[9] M. Varsta and J. Heikkonen (1997). Context learning with the self-organizing map, *Proc. Workshop on Self-Organizing Maps*, pp 197–202.

Extracting Finite Structure from Infinite Language

T. McQueen, A. A. Hopgood, T. J. Allen, and J. A. Tepper
School of Computing & Informatics, Nottingham Trent University,
Burton Street, Nottingham, NG1 4BU, UK
thomas.mcqueen{adrian.hopgood , tony.allen, jonathan.tepper}@ntu.ac.uk
www.ntu.ac.uk

Abstract

This paper presents a novel connectionist memory-rule based model capable of learning the finite-state properties of an input language from a set of positive examples. The model is based upon an unsupervised recurrent self-organizing map [1] with laterally interconnected neurons. A derivation of functional-equivalence theory [2] is used that allows the model to exploit similarities between the future context of previously memorized sequences and the future context of the current input sequence. This bottom-up learning algorithm binds functionally-related neurons together to form states. Results show that the model is able to learn the Reber grammar [3] perfectly from a randomly generated training set and to generalize to sequences beyond the length of those found in the training set.

1. Introduction

Since its inception, language acquisition has been one of the core problems in artificial intelligence. The ability to communicate through spoken or written language is considered by many philosophers to be the hallmark of human intelligence. Researchers have endeavoured to explain this human propensity for language in order both to develop a deeper understanding of cognition and also to produce a model of language itself. The quest for an automated language acquisition model is thus the ultimate aim for many researchers [4]. Currently, the abilities of many natural language processing systems, such as parsers and information extraction systems, are limited by a prerequisite need for an incalculable amount of manually derived language and domain-specific knowledge. The development of a model that could automatically acquire and represent language would revolutionize the field of artificial intelligence, impacting on almost every area of computing from Internet search engines to speech-recognition systems.

Language acquisition is considered by many to be a paradox. Researchers such as Chomsky argue that the input to which children are exposed is insufficient for them to determine the grammatical rules of the language. This argument for the poverty of stimulus [5] is based on Gold's theorem [6], which proves that most classes of languages cannot be learnt using only positive evidence, because of the effect of overgeneralization. Gold's analysis and proof regarding the unfeasibility of language acquisition thus forms a central conceptual pillar of modern linguistics. However, less formal approaches have questioned the treatment of language identification as a deterministic problem in which any solution must involve a guarantee of no future errors. Such approaches to the problem of language acquisition [7] show that certain classes of language can be learnt using only positive examples if language identification involves a stochastic probability of success.

Language acquisition, as with all aspects of natural language processing, traditionally involves hard-coded symbolic approaches. Such top-down approaches to cognition attempt to work backwards from formal linguistic structure towards human processing mechanisms. However, recent advances in cognitive modelling

have led to the birth of connectionism, a discipline that uses biologically inspired models that are capable of learning by example. In contrast to traditional symbolic approaches, connectionism uses a bottom-up approach to cognition that attempts to solve human-like problems using biologically inspired networks of interconnected neurons. Connectionist models learn by exploiting statistical relationships in their input data, potentially allowing them to discover the underlying rules for a problem. This ability to learn the rules, as opposed to learning via rote memorization, allows connectionist models to generalize their learnt behaviour to unseen exemplars. Connectionist models of language acquisition pose a direct challenge to traditional nativist perspectives based on Gold's theorem [6] because they attempt to learn language using only positive examples.

Connectionism and Determinacy

Since the early nineties, connectionist models such as the simple recurrent network (SRN) [8] have been applied to the language acquisition problem in the form of grammar induction. This involves learning simple approximations of natural language, such as regular and context-free grammars. These experiments have met with some success [6, 7], suggesting that dynamic recurrent networks (DRNs) can learn to emulate finite-state automata. However, detailed analysis of models trained on these tasks show that a number of fundamental problems exist that may derive from using a model with a continuous state-space to approximate a discrete problem.

While DRNs are capable of learning simple formal languages, they are renowned for their instability when processing long sequences that were not part of their training set [8, 9]. As detailed by Kolen [10], a DRN is capable of partitioning its state space into regions approximating the states in a grammar. However, sensitivity to initial conditions means that each transition between regions of state space will result in a slightly different trajectory. This causes instability when traversing state trajectories that were not seen during training. This is because slight discrepancies in the trajectories will be compounded with each transition until they exceed the locus of the original attractor, resulting in a transition to an erroneous region of state space. Such behavior is characteristic of continuous state-space DRNs and can be seen as both a power and a weakness of this class of model. While this representational power enables the model to surpass deterministic finite automata and emulate non-deterministic systems, it proves to be a significant disadvantage when attempting to emulate the deterministic behavior fundamental to deterministic finite state automata (DFA).

Attempts have been made to produce discrete state-space DRNs by using a step-function for the hidden layer neurons [9]. However, while this technique eliminates the instability problem, the use of a non-differentiable function means that the weight-update algorithm's sigmoid function can only approximate the error signal. This weakens the power of the learning algorithm, which increases training times and may cause the model to learn an incorrect representation of the DFA.

The instability of DRNs when generalizing to long sequences that are beyond their training sets is a limitation that is probably endemic to most continuous state-space connectionist models. However, when finite-state extraction techniques [9] are applied to the weight space of a trained DRN, it has been shown that once extracted into symbolic form, the representations learnt by the DRN can perfectly emulate the original DFA, even beyond the training set. Thus, while discrete symbolic models may be unable to adequately model the learning process itself, they are better suited to representing the learnt DFA than the original continuous state-space connectionist model.

While supervised DRNs such as the SRN dominate the literature on connectionist temporal sequence processing, they are not the only class of recurrent network. Unsupervised models, typically based on the self-organizing map (SOM) [11], have also been used in certain areas of temporal sequence processing [12]. Due to their localist nature, many unsupervised models operate using a discrete state-space and are therefore not subject to the same kind of instabilities characteristic of supervised continuous state-space DRNs. The aim of this research is therefore to develop an unsupervised discrete state-space recurrent connectionist model that can induce the finite-state properties of language from a set of positive examples.

A Memory-Rule Based Theory of Linguistics

Many leading linguists, such as Pinker [13] and Marcus [14], have theorized that language acquisition, as well as other aspects of cognition, can be explained using a memory-rule based model. This theory proposes that cognition uses two separate mechanisms that work together to form memory. Such a dual-mechanism approach is supported by neuro-biological research, which suggests that human memory operates using a declarative fact-based system and a procedural skill-based system [15]. In this theory, rote memorization is used to learn individual exemplars, while a rule-based mechanism operates to override the original memorizations in order to produce behaviour specific to a category. This memory-rule theory of cognition is commonly explained in the context of the acquisition of the English past tense [13]. Accounting for children's over-regularizations during the process of learning regular and irregular verbs constitutes a well-known battlefield for competing linguistic theories. Both Pinker [13] and Marcus [14] propose that irregular verbs are learnt via rote-memorization, while regular verbs are produced by a rule. The evidence for this rule-based behaviour is cited as the over-regularization errors produced when children incorrectly apply the past tense rule to irregular verbs (e.g. *runned* instead of *ran*).

The model presented in this paper is a connectionist implementation of a memory-rule based system that extracts the finite-state properties of an input language from a set of positive example sequences. The model's bottom-up learning algorithm uses functional-equivalence theory [2] to construct discrete-symbolic representations of grammatical states (Figure 1).

STORM (Spatio Temporal Self-Organizing Recurrent Map)

STORM is a recurrent SOM [1] that acts as a temporal associative memory, initially producing a localist-based memorization of input sequences. The model's rule-based mechanism then exploits similarities between the future context of memorized sequences and the future context of input sequences. These similarities are used to construct functional-relationships, which are equivalent to states in the grammar. The next two sections will detail the model's memorization and rule-based mechanisms separately.

STORM's Memorization Mechanism

STORM maintains much of the functionality of the original SOM [11], including the winning-neuron selection algorithm (Equation 1), weight-update algorithm (Equation 2) and neighbourhood function (Equation 3). The model's localist architecture is used to represent each element of the input sequence using a

separate neuron. In this respect, STORM exploits the SOM's abilities as a vector quantization system rather than as a topological map. Equation 1 shows that for every input to the model (X), the neuron whose weight vector has the lowest distance measure from the input vector is selected as the winning neuron (Y). The symbol *d* denotes the distance between the winning neuron and the neuron in question. As shown in fig 1, each input vector consists of the current input symbol and a context vector, representing the location of the previous winning neuron.

$$y_i = \arg\min_i(d(x, w_i)) \qquad (1)$$

The weight update algorithm (equation 2) is then applied to bring the winning neuron's weight vector (W), along with the weight vectors of neighbouring neurons, closer to the input vector (X) (equation 2). The rate of weight change is controlled by the learning rate α, which is linearly decreased through training.

$$w_{ij}(t+1) = w_{ij}(t) + \alpha h_{ij}(x(t) - w_{ij}(t)) \qquad (2)$$

The symbol *h* in equation 2 denotes the neighbourhood function (equation 3). This standard Gaussian function is used to update the weights of neighbouring neurons in proportion to their distance from the winning neuron. This weight update function, in conjunction with the neighbourhood function, has the effect of mapping similar inputs to similar locations on the map and also minimizing weight sharing between similar inputs. The width of the kernel σ is linearly decreased through training.

$$h_{ij} = \exp\left(\frac{-d_{ij}^2}{2\sigma^2}\right) \qquad (3)$$

The model uses an orthogonal input vector to represent the grammar's terminal symbols. Each of the seven terminal symbols are represented by setting the respective binary value to 1 and setting all the other values to 0 (table 1).

Grammatical symbol	Orthogonal vector
B	1 0 0 0 0 0 0
T	0 1 0 0 0 0 0
P	0 0 1 0 0 0 0
S	0 0 0 1 0 0 0
X	0 0 0 0 1 0 0
V	0 0 0 0 0 1 0
E	0 0 0 0 0 0 1

Table 1 – Orthogonal vector representations for input symbols

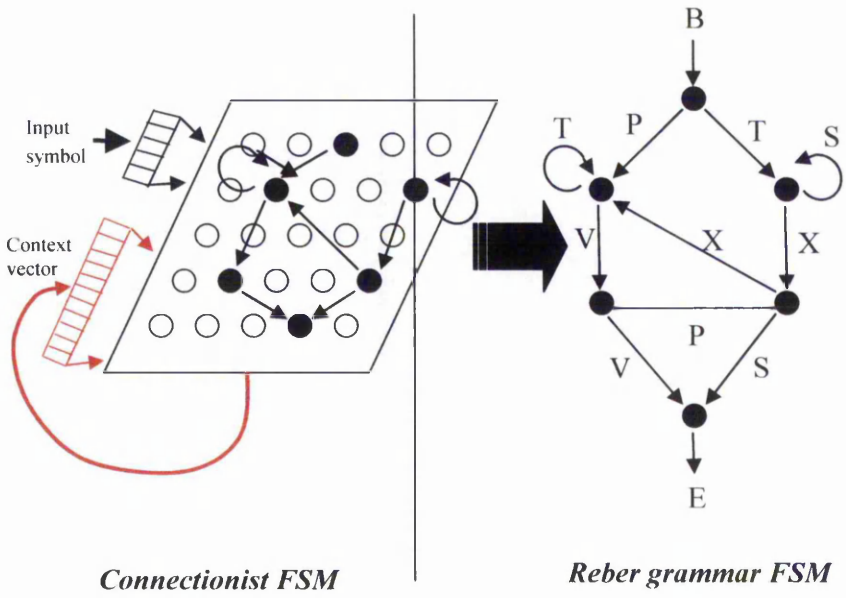


Fig. 1 – Diagram showing conceptual overview of model. The left side shows STORM's representation of a FSM, while the right side of the diagram shows the FSM for the Reber grammar.

As shown in Figures 1 and 2, STORM extends Kohonen's SOM [11] into the temporal domain by using recurrent connections. The recurrency mechanism feeds back a representation of the previous winning neuron's location on the map using a 10-bit Gray-code vector. By separately representing the column and row of the previous winning neuron in the context vector, the recurrency mechanism creates a 2D representation of the neuron's location. Further details of the recurrency mechanism, along with its advantages, are provided in [1]. This method of explicitly representing the previous winner's location as part of the input vector has the effect of selecting the winning neuron based not just on the current input, but also indirectly on all previous inputs in the sequence. The advantage of this method of recurrency is that it is more efficient than alternative methods (e.g. [16]), because only information pertaining to the previous winning neuron's location is fed back. Secondly, the amount of information fed back isn't directly related to the size of the map (i.e. recursive SOM [16] feeds back a representation of each neuron's activation). This allows the model to scale up to larger problems without exponentially increasing computational complexity.

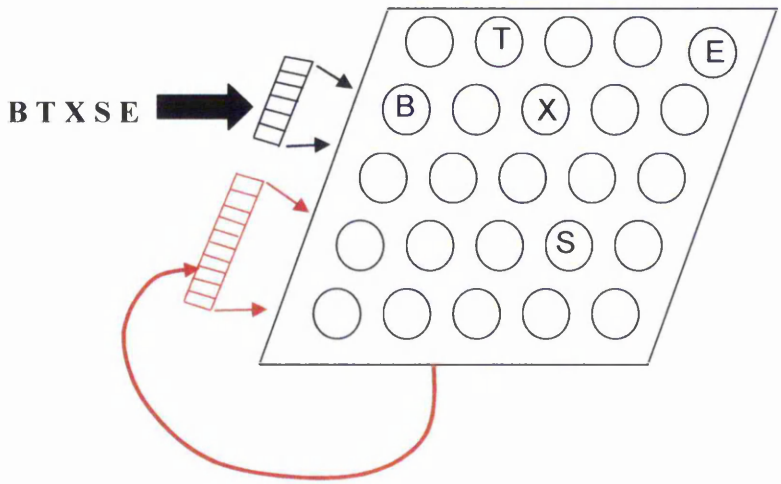


Fig. 2 – Diagram showing STORM's input representation. The model's weight vector consists of a 7-bit orthogonal symbol vector representing the terminal symbol in the grammar, along with a 10-bit Gray code context vector, representing the column and row of the previous winning neuron.

STORM's Rule-Based Construction Mechanism

The model's location-based recurrency representation and localist architecture provide it with a very important ability. Unlike using conventional artificial neural networks, the sequences learnt by STORM can be extracted in reverse order. This makes it possible to start with the last element in an input sequence and work backwards to find the winning neurons corresponding to the previous inputs in the sequence. STORM uses this ability, while processing input sequences, to find any existing pre-learnt sequences that end with the same elements as the current input sequence. For example, Figure 3 shows that the winning neuron for the symbol 'T' in sequence 1 has the same future context ('XSE') as the winning neuron for the first symbol 'S' in sequence 2.

Functional-equivalent theory [2] asserts that two states are said to be equivalent if, for all future inputs, their outputs are identical. STORM uses the inverse of this theory to construct states in a bottom-up approach to grammar acquisition. By identifying neurons with consistently identical future inputs, the model's temporal Hebbian learning mechanism (THL) mechanism binds together potential states via lateral connections. By strengthening the lateral connections between neurons that have the same future context, this THL mechanism constructs functional-relationships between the winning neuron for the current input and the winning neuron for a memorized input (referred to as the alternative winner) whose future-context matches that of the current input sequence (Figure 4). In order to prevent lateral weight values from becoming too high, a negative THL value is applied every time a winning neuron is selected. This has the effect of controlling lateral weight growth and also breaking down old functional relationships that are no longer used.

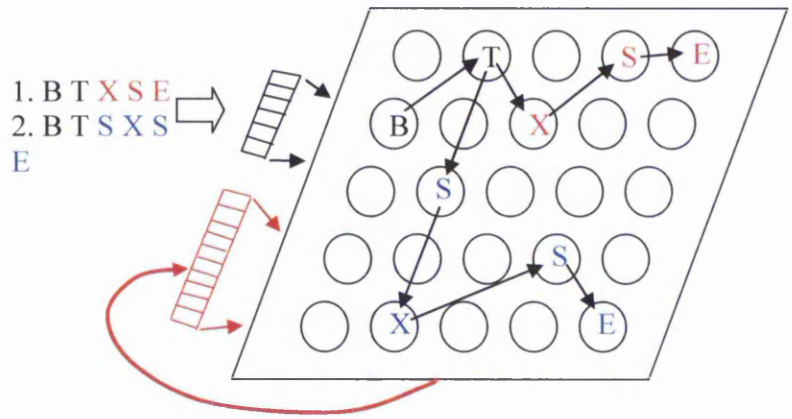


Fig. 3 – Diagram showing the memorized winning neurons for two sequences that end with the same sub-sequence 'XSE'

Once states have formed, they override the recurrency mechanism, forcing the model to use a single representation for the future inputs in the sequence rather than the original two representations (Figure 4). The advantage of forming states in this manner is that it provides the model with a powerful ability to generalize beyond its original memorizations. The model's THL mechanism conforms to the SOM's winner-take-all philosophy by selecting the alternative winner as the neuron whose future-context is the best match to that of the current input sequence. Given that tracing back through the future-context may identify multiple alternative winners, the criteria of best matching winner classifies the strongest sequence stored in the model as the winner. Furthermore, THL is only used to enhance the functional relationship between the winner and the alternative winner,

if the future-context for the alternative winner is stronger than that of the winner itself. Thus, the model has a preference for always using the dominant sequence and it will use the THL mechanism to re-wire its internal pathways in order to use any dominant sequence.

Constructing the lateral connections between functionally-related neurons is equivalent to identifying states in a grammar. Once the strength of these lateral connections exceeds a certain threshold they override the standard recurrency mechanism, affecting the representation of the previous winning neuron that is fed back (Figure 4). Instead of feeding back a representation of the previous winning neuron, the lateral connections may force the model to feed back a representation of the functionally-related neuron. The consequence of this is that the rest of the sequence is processed as if the functionally-related neuron had been selected rather than the actual winner. For example, Figure 4 shows that when the first 'S' symbol in sequence 2 is presented to STORM, its winning neuron is functionally linked to the winner for the 'T' symbol from sequence 1. As the latter winning neuron is the dominant winner for this state, its location is fed back as context for the next symbol in sequence 2.

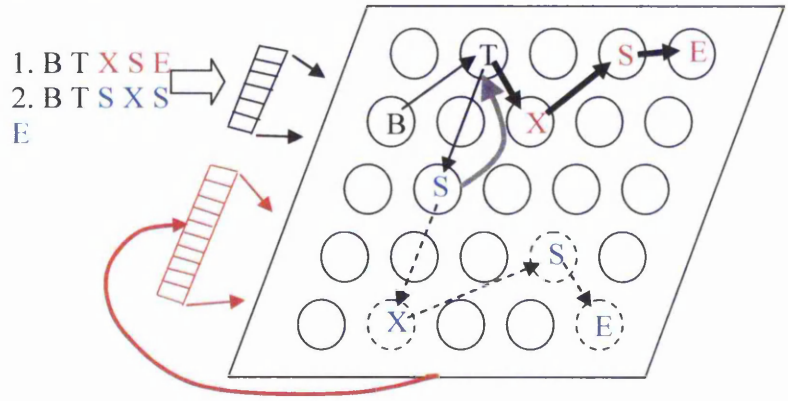


Fig. 4 – Functional override in winning-neuron selection algorithm. The functional relationship (shown in grey) between the third symbol 'S' in the second sequence and the second symbol 'T' in the first sequence, forces the model to process the remaining elements in the second sequence (namely 'XSE') using the same winning neurons as for the first sequence.

While a state is formed based on similarities in future context, there may be cases where the future context, for the respective input symbols that make up the state, is dissimilar (Table 2). However, once a state been constructed, the future context in subsequent sequences containing that state will be processed in an identical manner, regardless of the future context itself. For example, when trained on the sequences in Table 2, the 'T' symbol from sequence 1 will form a state with the first 'S' symbol from sequence 2. This will result in both sequences 1 and 2 sharing the same winning neurons for their final three inputs (X S E). STORM will then be able to generalize this learnt state to its memorization of sequence 3, resulting in the same winning neurons being activated for the 'X X V V E' in test sequence 4 as in training sequence 3.

#	Training sequence
1	B T X S E
2	B T S X S E
3	B T X X V V E

#	Test sequence
4	B T S X X V V E

Table 2 – Generalization example. When trained on the first three sequences, STORM is able to construct a state between the 'T' in sequence 1 and the first 'S' in sequence 2. By generalizing this learnt state to its memorization of sequence 3, STORM is able to correctly process sequence 4 by activating the same winning neurons for the sub-sequence 'X X V V E' as would be activated in sequence 3.

Experiments

In order to quantify STORM's grammar induction abilities, the model was applied to the task of predicting the next symbols in a sequence from the Reber grammar (Figure 1). Similar prediction tasks have been used in [8] and [3] to test the SRN's grammar-induction abilities. The task involved presenting the model with symbols from a randomly generated sequence that was not encountered during training. The model then had to predict the next possible symbols in the sequence that could follow each symbol according to the rules of the grammar. STORM's predictions are made by utilizing the locational representational values used in its context vector. As further explained in [1], the winning neuron for an input is the neuron whose weight vector best matches both the input symbol and the context representation of the last winning neuron's location. STORM predicts the next symbol by finding the neuron whose context representation best matches that of the current winning neuron (i.e. the symbol part of the weight vector is ignored in the Euclidean distance calculation). This forces the model to find the neuron that is most likely to be the next winner. The symbol part of this neuron's weight vector provides the next predicted symbol itself. This process is then repeated to find the second-best matching winner and the corresponding second predicted next symbol. In accordance with established training criteria for artificial neural network models [17], the experiments were conducted on randomly generated separate training and test sets (i.e. sequences were unique with respect to all other sequences in both sets). Such an approach ensures that the model's performance, assessed from the test set, is a true measure of its generalization abilities because the test sequences were not encountered during training. The experiment was run ten times using models with randomly generated initial weights, in order to ensure that the starting state did not adversely influence the results.

The recursive depth parameter, as listed in Table 3, denotes the maximum number of sequential recursive transversals a sentence may contain (i.e. how many times it can go around the same loop). In order to ensure that the training and test sequences are representative of the specified recursive depth, the sets are divided equally between sequences of each recursive depth (i.e. a set of six sequences with a recursive depth (RD) of 2 will contain two sequences with an RD of 0, two sequences with an RD of 1 and two sequences with an RD of 2).

Parameter	Value
Number of epochs	1000
Learning rate α (linearly decreasing)	0.1
Initial neighbourhood σ (linearly decreasing)	5
Positive / negative temporal Hebbian learning rate	0.5 / 0.005
Number of training sequences	21
Number of test sequences	7
Maximum recursive depth (RD) of sequences	6
Model size	10 × 10

Table 3 - Experimental parameters for the first experiment

As shown in figure 5, six models learnt the grammar with over 89% accuracy during training and three of them became perfect grammar recognizers. However, this number fell by the end of training, with only two perfect models and an additional two models with over 90% performance accuracy. This equates to an average post-training performance of 71%. While less than half the models successfully learnt the grammar, it is worth noting that this is significantly better than for SRNs where Sharkey [18] showed that only two out of 90 SRNs became finite-state grammar recognisers in a similar experiment using the Reber grammar.

One of the proposed advantages of a discrete state-space model (page 3), is its ability to generalize to sequences longer than those encountered during training without the instabilities characteristic of standard DRN models. In order to test this proposition, a perfect finite-state recognizer (i.e. a model that scored 100% prediction accuracy) from the first experiment (figure 5) was tested on a further three test sets. These sets contained sequences with recursive depths of 8, 10 and 12 and should constitute a much harder problem for any model trained only on sequences with a recursive depth of 6. These models that achieved 100% performance accuracy in the original experiments also achieved 100% accuracy on training sets with higher recursive depths. This proves that these models act as perfect grammar recognizers that are capable of generalizing to sequences of potentially any length.

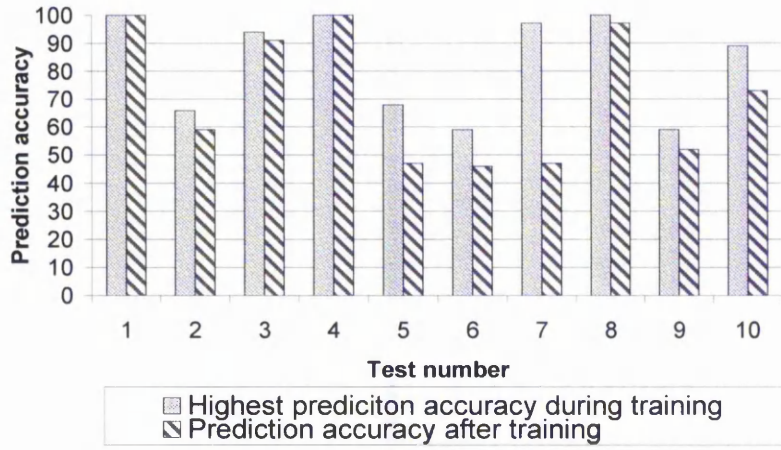


Fig 5 – Results from ten models trained on randomly generated separate training and test sets.

Conclusions and Future Work

We have presented a novel connectionist memory-rule based model capable of inducing the finite-state properties of an input language from a set of positive example sequences. In contrast with the majority of supervised connectionist models in the literature, STORM is based on an unsupervised recurrent SOM [1] and operates using a discrete state-space.

The model has been successfully applied to the task of learning the Reber grammar by predicting the next symbols in a set of randomly generated sequences. The experiments have shown that over half the models trained are capable of learning a good approximation of the grammar (over 89%) during the training process. However, by the end of training, only a fifth of the models were capable of operating as perfect grammar recognizers. This suggests that the model is unstable and that partial or optimal solutions reached during training may be lost by the end of the training process. Despite this instability, a comparison between STORM and the SRN, when applied to a similar problem [3], shows that STORM is capable of learning the grammar perfectly much more often than its counterpart. Furthermore, experiments show that STORM's discrete state-space allow it to generalize its grammar recognition abilities to sequences far beyond the length of those encountered in the training set, without the instabilities experienced in continuous state-space DRNs.

Future work will initially involve analyzing the model to find where it fails. Once the model's abilities have been fully explored, its stability will be improved to increase the number of models that successfully become perfect grammar recognizers. STORM will then be enhanced to allow it to process more advanced grammars. Given that regular grammars are insufficient for representing natural language [19], the model must be extended to learn at least context-free languages if it is to be applied to real-world problems. However, despite such future requirements STORM's current ability to explicitly learn the rules of a regular grammar distinguish its potential as a language acquisition model.

References

1. McQueen, T. & Hopgood, A. & Tepper, J. & Allen, T. A Recurrent self-organizing map for Temporal Sequence Processing. In: Proceedings of 4th International Conference in Recent Advances in Soft Computing (RASC2002), Nottingham, 2002
2. Hopcroft J. & Ullman J. Introduction to Automata Theory, Languages and Computation, vol 1, Addison-Wesley, 1979
3. Cleeremans A, Schreiber D, McClelland J. Finite State Automata and Simple Recurrent Networks. In: Neural Computation. 1989, Vol 1, pp 372-381
4. Collier R. An historical overview of natural language processing systems that learn. Artificial Intelligence Review 1994; 8(1)
5. Chomsky, N. Aspects of the Theory of Syntax. MIT Press, 1965
6. Gold, E.M. Language Identification in the Limit. Information and Control 1967; 10:447-474
7. Horning, J.J. A study of grammatical inference. PhD thesis, Stanford University, California, 1969
8. Elman, J.L. Finding Structure in Time. Cognitive Science 1990; 14:179-211
9. Omlin, C. Understanding and Explaining DRN Behaviour. In: Kolen, J. and Kremer S (eds) A Field Guide to Dynamical Recurrent Networks. IEEE Press, New York, 2001, pp 207-227
10. Kolen, J. Fool's Gold: Extracting Finite State Machines From Recurrent Network Dynamics. In: Cowan J, Tesauro G and Alspector J (eds) Advances in Neural Information Processing Systems 6. Morgan Kaufmann, San Francisco CA, 1994, pp 501-508
11. Kohonen T. Self-Organizing Maps, vol 1. Springer-Verlag, Germany, 1995
12. Baretto, G and Arajo, A. Time in Self-Organizing Map: An Overview of Models. International Journal of Computer Research: Special Edition on Neural Networks: Past, Present and Future 2001; 10(2):139-179
13. Pinker, S. Words and Rules. Phoenix, London, 2000
14. Marcus, G. F. Children's Overregularization and Its Implications for Cognition. In: P. Broeder and J. Murre (eds) Models of Language Acquisition: Inductive and Deductive approaches. Oxford University Press, Oxford, 2000, pp 154-176
15. Cohen, N.J. and Squire, L.R. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that. Science 1980; 21:207-210
16. Voegtlin, T. Recursive Self-Organizing Maps. Neural Networks 2002; 15(8-9):979-991
17. Hopgood, A. A. Intelligent Systems for Engineers and Scientists, 2nd edition, CRC Press LLC, Florida, 2001, pp 195-222
18. Sharkey N, Sharkey A, Jackson S. Are SRNs sufficient for modelling language acquisition?. In: Broeder P, Murre J. (eds) Models of Language Acquisition: Inductive and Deductive Approaches. Oxford University Press, Oxford, 2000, pp 33-54
19. Lawrence S, Giles C, Fong S. Natural Language Grammatical Inference with Recurrent Neural Networks. IEEE Transactions on Knowledge and Data Engineering 2000; 12(1):126-140

Appendix D – Examples of training and test sequences

The following sequences have been used to train a model to 100% performance on the Reber grammar using the training regime described in section 4.5.

The following sequences are in their numerical form as used during training. The numbers can be translated in grammatical symbols using the following key.

Key: B = 1, T = 2, S = 4, X = 5, V = 6, P = 3, E = 7

Training Sequences:

```
1 2 4 4 4 4 5 5 6 6 7
1 2 4 5 5 2 6 3 5 2 2 2 2 6 6 7
1 2 5 5 6 3 5 6 6 7
1 3 2 6 3 5 6 6 7
1 2 4 4 5 5 6 3 4 7
1 2 4 5 5 6 6 7
1 2 5 4 7
1 2 5 5 2 2 2 2 2 2 6 6 7
1 2 4 5 4 7
1 3 6 3 5 2 2 2 2 2 6 3 5 2 2 2 2 6 3 5 6 3 5 6 6 7
1 3 6 3 4 7
1 3 6 6 7
1 2 5 5 2 6 6 7
1 2 5 5 6 3 5 6 3 4 7
1 2 4 5 5 2 6 6 7
1 3 2 6 3 4 7
1 2 4 5 5 6 3 5 6 6 7
1 3 6 3 5 2 2 6 6 7
1 3 2 2 2 2 2 2 6 6 7
1 2 5 5 6 6 7
1 2 5 5 6 3 4 7
1 2 4 5 5 2 2 6 6 7
1 3 6 3 5 6 3 5 2 6 3 5 2 2 6 6 7
1 3 2 6 6 7
1 2 4 4 4 5 4 7
1 2 5 5 6 3 5 2 6 3 4 7
1 2 4 4 5 4 7
1 2 4 4 5 5 6 6 7
1 2 4 5 5 6 3 5 2 6 3 4 7
1 2 4 5 5 6 3 4 7
```

Test sequences:

```
1 2 5 5 2 6 3 4 7
1 2 5 5 2 2 6 6 7
1 2 4 4 5 5 6 3 5 6 3 4 7
1 2 4 4 4 4 4 4 5 5 2 6 6 7
1 3 2 6 3 5 2 2 2 6 3 4 7
1 2 5 5 2 2 2 2 6 6 7
```

```
1 2 4 4 5 5 2 2 2 6 3 4 7
1 3 2 2 2 6 6 7
1 3 2 2 6 3 5 6 6 7
1 2 4 5 5 6 3 5 2 6 6 7
```

Libraries &
Learning
Resources

The Boots Library: 0115 848 6343
Clifton Campus Library: 0115 848 6612
Brackenhurst Library: 01636 817049