ELSEVIER
Available online at www.sciencedirect.com
SCIENCE DIRECT
Knowledge-Based Systems xx (xxxx) 1–7
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

Extracting finite structure from infinite language

T. McQueen, A.A. Hopgood, T.J. Allen, J.A. Tepper*

School of Computing and Informatics, Nottingham Trent University, Burton Street, Nottingham NG1 4BU, UK

Received 26 October 2004; accepted 30 October 2004

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 [T. McQueen, A. Hopgood, J. Tepper, T. Allen, A recurrent self-organizing map for temporal sequence processing, in: Proceedings of Fourth International Conference in Recent Advances in Soft Computing (RASC2002), Nottingham, 2002] with laterally interconnected neurons. A derivation of functional-equivalence theory [J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages and Computation, vol. 1, Addison-Wesley, Reading, MA, 1979] 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 [A. Cleeremans, D. Schreiber, J. McClelland, Finite state automata and simple recurrent networks, Neural Computation, 1 (1989) 372–381] perfectly from a randomly generated training set and to generalize to sequences beyond the length of those found in the training set.

Keywords: Artificial neural networks; Grammar induction; Natural language processing; Self-organizing map; STORM

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 [5]. 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 [2] is based on Gold's theorem [7], 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 [10] show that certain classes of language can be learnt using only positive examples if language identification involves a stochastic probability of success.

* Corresponding author. Tel.: +44 1604 779 109; fax: +44 1604 716 247.

E-mail addresses: thomas.mcqueen@ntu.ac.uk (T. McQueen), adrian.hopgood@ntu.ac.uk (A.A. Hopgood), tony.allen@ntu.ac.uk (T.J. Allen), jonathan.tepper@ntu.ac.uk (J.A. Tepper).

URL: www.ntu.ac.uk.

0950-7051/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.knosys.2004.10.010

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 [7] because they attempt to learn language using only positive examples.

2. Connectionism and determinacy

Since the early nineties, connectionist models such as the simple recurrent network (SRN) [6] 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 [12], 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 behaviour 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 behaviour 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 [16]. 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 [16] 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 [1]. 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.

3. A Memory-rule based theory of linguistics

Many leading linguists, such as Pinker [17] 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 [4]. 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 [17]. Accounting for children's over-regularizations during the process of learning regular and irregular verbs constitutes

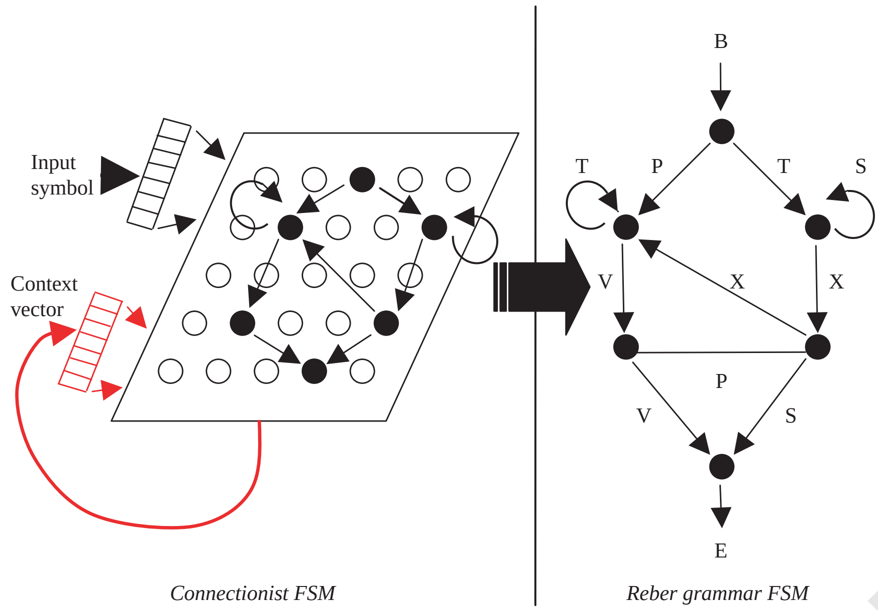

Fig. 1. Diagram showing conceptual overview of model. The left side shows STORM's representation of a FSM, while the right side of the diagram shows the FSM for the Reber grammar.

a well-known battlefield for competing linguistic theories. Both Pinker [17] 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 [8] to construct discrete-symbolic representations of grammatical states (Fig. 1).

4. STORM (Spatio Temporal Self-Organizing Recurrent Map)

STORM is a recurrent SOM [15] 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.

4.1. STORM's memorization mechanism

As shown in Figs. 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 [15].

The 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. [19]), because only information pertaining to the previous winning neuron's location is fed back. Secondly, the amount of information fed back is not directly related to the size of the map (i.e. recursive SOM [19] feeds back a representation of

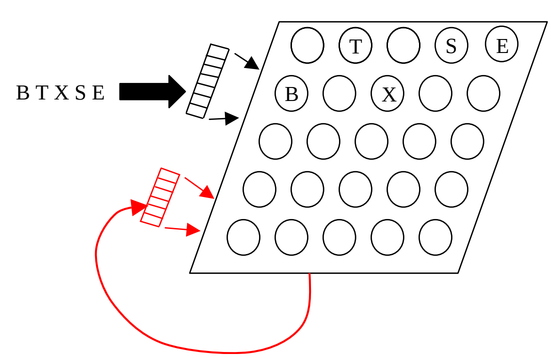

Fig. 2. Diagram showing STORM's input representation. The model's weight vector consists of a 7-bit orthogonal symbol vector representing the terminal symbol in the grammar, along with a 6-bit Gray code context vector, representing the column and row of the previous winning neuron.

Table 1
Orthogonal vector representations for input symbols

Grammatical symbol	Orthogonal vector
B	1 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

each neuron's activation). This allows the model to scale up to larger problems without exponentially increasing computational complexity.

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). For example, in Fig. 2 the input vector that would be applied when the symbol 'T' is presented would be 0 1 0 0 0 0 0 0 0 1 1 1 0. The first 7 bits of this vector represent the input symbol, while the remaining six bits represent the context vector.

STORM maintains much of the functionality of the original SOM [11], including the winning-neuron selection algorithm (Eq. (1)), weight-update algorithm (Eq. (2)) and neighbourhood function (Eq. (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. Eq. (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. The winning neuron is, therefore, the neuron whose overall weight vector is the closest to the combination of the input symbol vector and context vector.

$$y_i = \arg\min_i(d(x, w_i)) \tag{1}$$

The weight-update algorithm (Eq. (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) (Eq. (2)). The rate of weight change is controlled by the learning rate α, which is linearly decreased through training.

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

The symbol h in Eq. (2) denotes the neighbourhood function (Eq. (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.

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

4.2. STORM's rule-based construction mechanism

The model's location-based recurrency representation and localist architecture provide it with a very important ability. The sequences learnt by STORM, unlike conventional artificial neural networks, 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 any stored 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, Fig. 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 [8] 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 (Fig. 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

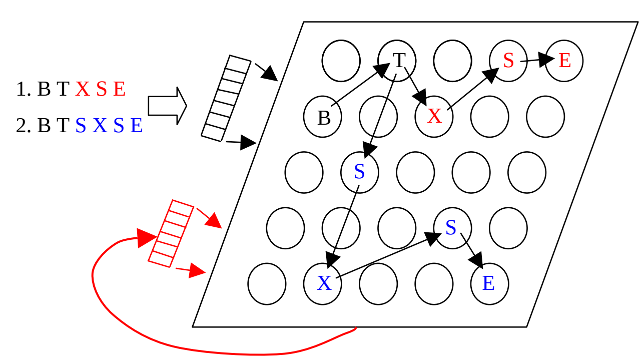

Fig. 3. Diagram showing the memorized winning neurons for two sequences that end with the same sub-sequence 'XSE'.

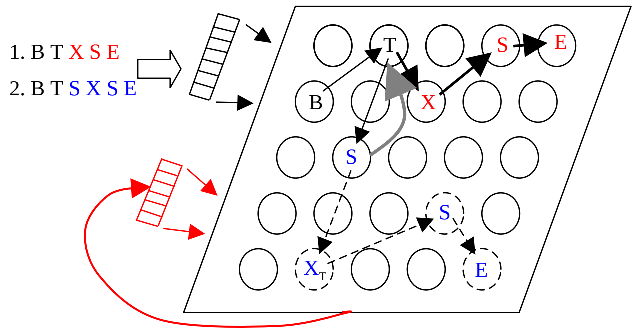

Fig. 4. Functional override in winning-neuron selection algorithm. The functional relationship (shown in grey) between the third symbol ‘S’ in the second sequence and the second symbol ‘T’ in the first sequence, forces the model to process the remaining elements in the second sequence (namely ‘XSE’)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

growth and also breaking down old functional relationships that are no longer used.

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 (Fig. 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 (Fig. 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, Fig. 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.

Table 2
Generalization example

#	Training sequence symbols/number of corresponding winning neuron
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
4	B T S X X V V E
	4 10 8 14 2 12 18 23

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.

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.

5. 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 (Fig. 1). Similar prediction tasks have been used in [6] 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 [15], 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

Table 3
Experimental parameters for the first experiment

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

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 [9], 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).

As shown in Fig. 5, six models learnt the grammar with over 89% accuracy during training and three of them became perfect grammar recognizers. However, this

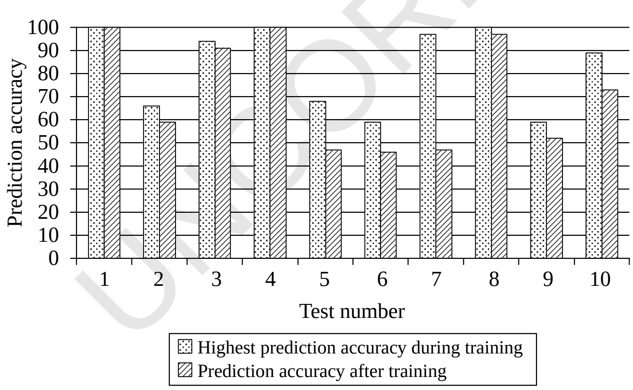

Fig. 5. Results from ten models trained on randomly generated separate training and test sets.

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 (Fig. 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.

6. 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 [15] 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 involve analyzing the model to find, where it fails. Once the model's abilities have been 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 [13], 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] G. Baretto, A. Arajo, Time in self-organizing maps: an overview of models, International Journal of Computer Research: Special Edition on Neural Networks: Past, Present and Future 10 (2) (2001) 139–179.

[2] N. Chomsky, Aspects of the Theory of Syntax, MIT Press, Cambridge, MA, 1965.

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

[4] N.J. Cohen, R. Squire, Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that, Science 210 (1980) 207–210.

[5] R. Collier, An historical overview of natural language processing systems that learn, Artificial Intelligence Review 8 (1) (1994).

[6] J.L. Elman, Finding structure in time, Cognitive Science 14 (1990) 179–211.

[7] E.M. Gold, Language identification in the limit, Information and Control 10 (1967) 447–474.

[8] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages and Computation, vol. 1, Addison-Wesley, Reading, MA, 1979.

[9] A.A. Hopgood, Intelligent Systems for Engineers and Scientists, 2nd ed., CRC Press LLC, Florida, 2001. pp. 195–222.

[10] J.J. Horning, A study of grammatical inference, Ph.D thesis, Stanford University, California, 1969.

[11] T. Kohonen, Self-Organizing Maps, vol. 1, Springer, Germany, 1995.

[12] J. Kolen, Fool's gold: extracting finite state machines from recurrent network dynamics in: J. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems vol. 6, Morgan Kaufmann, San Francisco CA, 1994, pp. 501–508.

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

[14] G.F. Marcus, Children's overregularization and its implications for cognition in: P. Broeder, J. Murre (Eds.), Models of Language Acquisition: Inductive and Deductive approaches, Oxford University Press, Oxford, 2000, pp. 154–176.

[15] T. McQueen, A. Hopgood, J. Tepper, T. Allen, A recurrent self-organizing map for temporal sequence processing, in: Proceedings of Fourth International Conference in Recent Advances in Soft Computing (RASC2002), Nottingham, 2002.

[16] C. Omlin, Understanding and explaining DRN behaviour in: J. Kolen, S. Kremer (Eds.), A Field Guide to Dynamical Recurrent Networks, IEEE Press, New York, 2001, pp. 207–227.

[17] S. Pinker, Words and Rules: The Ingredients of Language, Phoenix, London, 2000.

[18] N. Sharkey, A. Sharkey, S. Jackson, Are SRNs sufficient for modelling language acquisition? in: P. Broeder, J. Murre (Eds.), Models of Language Acquisition: Inductive and Deductive Approaches, Oxford University Press, Oxford, 2000, pp. 33–54.

[19] T. Voegtlin, Recursive self-organizing maps, Neural Networks 15 (8–9) (2002) 979–991.