Parsing recursive sentences with a connectionist model including a neural stack and synaptic gating

Anna Fedor, Péter Péter Ittzés, Eörs Eörs Szathmáry

Parsing recursive sentences with a connectionist model including a neural stack and synaptic gating

Anna Fedor, Péter Ittzés, Eörs Szathmáry

PII: S0022-5193(10)00621-1
DOI: doi:10.1016/j.jtbi.2010.11.026
Reference: YJTBI 6248

To appear in: *Journal of Theoretical Biology*

Received date: 21 July 2010
Revised date: 21 November 2010
Accepted date: 22 November 2010

Parsing recursive sentences with a connectionist model including a neural stack and synaptic gating

Anna Fedor[a,b,*] (fedoranna@gmail.com)
Péter Ittzés[b] (ittzes@gmail.com)
Eörs Szathmáry[a,b,c] (szathmary@colbud.hu)

[a] *Institute of Biology, Eötvös Loránd University*
1/C Pázmány Péter stny, 1117, Budapest, Hungary
tel.: 00361-3812187; fax: 00361-3812188

[b] *Collegium Budapest (Institute for Advanced Study)*
2 Szentháromság utca, 1014, Budapest, Hungary
tel.: 00361-2248300; fax: 00361-2248310

[c] *Parmenides Center for the Study of Thinking*
Munich, Germany

** Corresponding author*

Abstract

It is supposed that humans are genetically predisposed to be able to recognize sequences of context free grammars with center-embedded recursion while other primates are restricted to the recognition of finite state grammars with tail-recursion. Our aim was to construct a minimalist neural network that is able to parse artificial sentences of both grammars in an efficient way without using the biologically unrealistic backpropagation algorithm. The core of this network is a neural stack-like memory where the push and pop operations are regulated by synaptic gating on the connections between the layers of the stack. The network correctly categorizes novel sentences of both grammars after training. We suggest that the introduction of the neural stack memory will turn out to be substantial for any biological 'hierarchical processor' and the minimalist design of the model suggests a quest for similar, realistic neural architectures.

Keywords:
synaptic gating; neural stack; recursion; context-free grammar; finite state grammar

Introduction

Natural language is a fascinating phenomenon, very much in the focus of various disciplines, from linguistics proper to evolutionary biology (Bickerton, 1990; Hauser et al., 2002; Hurford, 2007; Maynard Smith and Szathmáry, 1995; Pinker, 1994). Although there is no general agreement on how to best characterize language, let alone its biological foundations, we follow the view that it is based on the critical combination of symbolic reference with complex syntax (Szathmáry, 2007). A crucial element of syntax is recursion (Corballis, 2007a; Hauser et al., 2002). Two main types of recursion occurring in natural language are tail- or end-recursion (including left and right-branching recursion) and centre-embedded recursion (CER). An example of left-branching tail-recursion is (after the popular British nursery rhyme *The house that Jack built*):

(1) The rat squeaked.

(2) The cat killed the rat that squeaked.

(3) The dog worried the cat that killed the rat that squeaked.

In these cases, sentences are characterized by the concatenation of coherent noun-verb pairs, and can be produced or parsed by simple iteration (Christiansen and Chater, 1999). Iteration is present in animal calls, like that of primates (e.g., (Robinson, 1984; Zuberbühler, 2002)) and songbirds (Eens, 1997). It was also proved that some animal species are able to infer the iterative rule from samples of artificial strings and generalize over novel strings (Robinson, 1984). If we represented word-pairs only, where word-pairs consist of two words with dependency between them (e.g., in sentence 3, these are: *dog-worried*, *cat-killed* and *rat-squeaked*), such sentences composed of three word-pairs can be described by the following rule:

(4) A_3 B_3 A_2 B_2 A_1 B_1,

where As represent nouns, Bs represent verbs, and words with the same index form word-pairs.

The above sentences (2 and 3) can be transformed to have centre-embedded structure:

(5) The rat that the cat killed squeaked.

(6) The rat that the cat that the dog worried killed squeaked.

Here, the general rule for three word-pairs is:

(7) A_1 A_2 A_3 B_3 B_2 B_1.

CER is claimed to be a general human capacity whereas it cannot be found in animal communication systems (Fitch and Hauser, 2004). It has not been proved either that any animal species is capable of learning CER in the laboratory. CER can be parsed by context-free grammar (CFG), which has higher generative power than finite-state grammar (FSG) which basically concatenates items (applies tail-recursion, (Corballis, 2007a); for the comprehensive hierarchy of formal grammars, see (Chomsky, 1957)).

In natural languages, every noun has several possible verb pairs, and vice versa, every verb has several possible noun pairs. In the above examples semantic relationship connects the words: usually only rats squeak, not dogs or cats. However, if the other two word-pairs are swapped, the sentence still makes sense:

(8) The dog killed the cat that worried the rat that squeaked.

In a slightly modified version of the sentence, it is not possible to swap the word-pairs, because singular and plural words must be matched:

(9) Dogs worry the cat that kills rats that squeak.

As a result of this grammatical constraint (together with the semantic constraint), even if the words were mixed without grammatical structure, it would be easy to see the coherent noun-verb pairs. In artificial languages without semantics, if these dependencies between words are not established somehow, sentences could be represented by a simpler structure, A^nB^n, e.g. for n=3:

(10) A A A B B B.

This grammar is called counting recursion (Christiansen and Chater, 1999) because parsing of this kind of sentences is possible by counting As and Bs. If the number of As and Bs is equal and there is only one transition from As to Bs the sentence is correct (Corballis, 2007a; Corballis, 2007b).

The structures that word-pairs imply are shown on Figure 1. In the case of tail-recursion, members of word-pairs are next to each other connected by local dependencies. Additionally, in sentences with CER there are word-pairs whose members are separated by other word-pairs thus they have long-distance (or long-range) dependencies. This implies a hierarchical structure compared to the linear structure of sentences with tail-recursion. The more levels this hierarchical structure has, the more words have to be remembered to be able to parse these sentences. In a six-word-long sentence with CER, the maximum number of words that has to be stored in memory is three and the first word has to be remembered until the presentation of the last one. In sentences with tail-recursion, members of word-pairs are presented shortly after each other, hence there is only one word that has to be remembered at a time. In counting recursion no individual word has to be stored in memory for grammaticality judgement, just the category of words passed and their quantity has to be remembered until the end of the sentence.

Some confusion resulted from using sentences of counting recursion in artificial language learning experiments and not differentiating them clearly from the more complex centre-embedded sentences. Fitch and Hauser (Fitch and Hauser, 2004) claimed that their human subjects were able to learn both FSG and CFG in a small artificial language, whereas cotton-top tamarins could learn only FSG. Since they did not establish dependencies between words their CFG sentences could be parsed by counting recursion. Likewise, Gentner, Fenn, Margoliash & Nusbaum (Gentner et al., 2006) claimed that their subjects (starlings) learnt CER using the same kind of structures as Fitch and Hauser (Fitch and Hauser, 2004). This in turn elicited some critique: Corballis (Corballis, 2007a; Corballis, 2007b) called attention to the fact that the sentences used could be parsed by simple counting, while others (De Vries et al., 2008; Perruchet and Rey, 2005) showed that in experimental situation similar to that of Fitch & Hauser (Fitch and Hauser, 2004) even human subjects used alternative strategies to solve the tasks.

Despite the controversy in artificial language experiments, there is a general agreement that centre-embedded structures are present in natural human languages, but not in natural animal communication systems. The question is, why. A simple answer would be that animals do not have Universal Grammar, but this leaves completely in the dark what the relevant biological differences could be. In fact there are serious doubts on the idea that abstract rules of Universal Grammar could ever get assimilated in the genome (Deacon, 2003; Wiles et al., 2005) but, in contrast, it cannot be doubted that some language-related genetic differences between humans and animals do exist. It is perhaps much more rewarding to enquire about the possible neuronal operations (procedures) that the brain could implement in order to handle language.

Connectionist models have increasingly been used to model empirical data across many areas of language processing (Christiansen and Chater, 2001a). However, connectionist models aiming at parsing recursive structures are often restricted to counting recursion. Sun et al.

(Sun et al., 1998) implemented a hybrid system, in which a recurrent neural network was coupled to an external non-neural stack memory. After training with backpropagation, the system was able to infer a CFG from input. In another study, continuous-time recurrent networks without a stack can learn both context-free and context-sensitive languages in a prediction task, using backpropagation through time (Bodén and Wiles, 2000). Since there were no long-range dependencies connecting words within the sentences, performance of these systems boiled down to counting (Rodriguez et al., 1999).

Other studies used input data conforming to real CER (as opposed to counting recursion) to train artificial neural networks. Elman (Elman, 1991) trained a simple recurrent network (SRN) on multiclausal sentences which contained multiply-embedded relative clauses. The network achieved a high level of performance in predicting the next word in the sentences. In a related model Christiansen and Chater (Christiansen and Chater, 1999) trained SRNs on recursive artificial languages. The behaviour of these networks was similar to human performance in that they reached higher performance in right-branching structures than in centre-embedded structures. In both studies backpropagation of error was used as a learning algorithm which is generally considered biologically implausible because it requires passage of information backward through synapses and along axons and because it uses error signals that must be precise and different for each neuron in the network (Mazzoni et al., 1991; O'Reilly, 1996).

Handling of hierarchical structures occurs at high speed during language production and comprehension, and it seems reasonable to assume that it requires specialized neural networks to do so (Fedor et al., 2009). It is well known that parsing of CER can be solved very efficiently by a stack (push-down automaton), with the necessary pop and push operations (Hopcroft and Ullman, 1979). Thus it would be a step forward to present a neurally plausible simple stack architecture that could parse CER. Along this line, Chen and Honavar (Chen and Honavar, 1999) proposed an artificial neural network architecture for syntax analysis which is assembled from neural network components for lexical analysis, stack, parsing and parse tree construction. The stack in their model is a fairly complex system that is composed of five different modules that have specifically designed connections and the stack requires four sets of binary inputs. We aimed at constructing a more minimalist architecture for a neural stack that is more similar to a push-down automaton in its architecture.

In this paper we will present a neural network which can be trained to parse sentences with tail-recursion and CER. Three key features of the model are: a) absence of backpropagation, b) a crucial role for synaptic gating, c) and a neurally implemented stack. These components of our model are not new, but the combination of these features is unprecedented – this is what makes this model very effective and minimalistic.

Methods

Grammars

We composed input sentences according to two types of recursion, namely tail-recursion and CER. Words were 0/1 binary strings, where there was only one 1 in each word (all the other digits are 0). Words were randomly divided into two groups, A and B. Each word from group A had exactly one (randomly chosen) pair from group B, and vice versa. Sentences (4) and (7) give examples for six-word-long sentences with tail-recursion and CER, respectively. Since no word occurs twice in a sentence, 8*7*6=336 sentences could be generated for each grammar.

Additionally, random agrammatical sentences were also generated. These sentences were also composed of three A words and three B words and always started with an A, just as grammatical sentences, but did not conform to any of the above rules.

Architecture and functioning of the network after successful training

The neural network consists of the following main modules: input layer, stack, predictor, two push-pop neurons and a decision neuron (Fig. 2.). The input layer receives one word at a time from the sentence. In the case of a 16-word vocabulary, the input layer has 16 units (neurons), where each unit corresponds to a single word. Second, there is a clocked stack (for a clocking mechanism see (Hjelmfelt et al., 1991)) with three layers, where every layer of the stack consists of 16 neurons. Adjacent neurons within a column of the stack are connected bidirectionally to each other, with a weight of 1. The third component, called the predictor, tries to predict the next word in the sentence based on the word that is stored in the top layer of the stack. The push-pop neurons have input connections from the predictor and the input layer. They basically compare the two, and if they store the same words (which means that the prediction was correct) signal +1, if they store different words (or if the predictor is empty) signal -1. The output connections of the push-pop neurons perform neural gating on the synapses of the stack: these connections modulate the synapses directly by enhancing or inhibiting them (i.e., the push-pop neurons are the so called ‘gatekeepers’). One of the push-pop neurons is an excitatory neuron which is connected to each upward synapse in the stack with positive weights (i.e., gating acts in a permissive fashion (Katz, 2003)) and the other one is an inhibitory neuron, which is connected to each downward synapse in the stack with negative weights (absolute suppressive gating). The signal that arrives to a synapse from a gating neuron is the product of the activation of the gating neuron and weight on the synapse of the gating neuron, just like with any other neurons. The difference is that negative signal from a gating neuron blocks the synapse it is connected to, while positive signal from a gating neuron makes it possible for the synapse to work. As a result, if the prediction was correct and the push-pop neurons signal +1, downward connections will be inhibited and upward connections will be enhanced in the stack, hence upward connections will predominate, and each layer will take the value of the layer bellow it (a pop action). In this case, the bottom layer becomes empty. On the other hand, if the prediction was not correct and the push-pop neurons signal -1, upward connections will be inhibited and downward connections will be enhanced in the stack, such that the downward synapses will predominate and every layer will take the value of the layer above it (a push action). In this case, the top layer takes its value from the input. Lastly, there is a decision neuron, which is connected to the top layer of the stack and signals only if there is a word stored on the top layer of the stack. The signalling of this neuron can be considered as the decision of the network on the grammaticality of the sentence: signalling means that the sentence encountered so far was ungrammatical, while 0 output means that the sentence is grammatical.

Now, let us see how the whole network with a three-layer-deep stack is supposed to parse a six-word-long sentence after learning. First, the predictor tries to predict the first word from the top of the stack, but since the stack is empty at the beginning, the predictor will have no prediction. Next, the first word arrives to the input layer and then the push-pop neurons compare the input with the prediction. Since prediction is unsuccessful (there is no word on the predictor), the push-pop neurons perform a push action on the stack and the top layer of the stack becomes occupied by the first word. This triggers the decision neuron, which will signal that the string encountered so far is ungrammatical. In the case of a sentence with tail-recursion, the next word depends on the previous one. At the beginning of the next cycle, the predictor predicts the next word based on the previous word which is stored on the top of the stack. If the prediction is correct, the push-pop neurons will perform a pop action on the stack, which will become empty again. This will suppress the signalling of the decision neuron which means that the sentence encountered so far is grammatical. The same push-pop actions are repeated with the next two word-pairs until the end of the sentence. To measure the performance of the network we can detect its predictions for the following words or its decisions on the grammaticality of the sentence. During the processing of a sentence with tail-recursion, the network is able to predict every second word (other words can not be predicted, hence maximum performance is 50%) and decides that the sentence is incorrect three times: after the first, third, and fifth word. Note, that independently of the length of the sentence, substrings of a grammatical sentence with even number of words are in fact grammatical in the case of tail-recursion.

In the case of an agrammatical sentence, one or more words do not have a pair, which means that more than three words are unpredictable in the sentence. This results in more than three push actions, which means that the stack is not empty at the end of the sentence and the decision neuron will signal that the sentence was agrammatical.

Parsing grammatical sentences with CER also involves equal number of push and pop actions; hence the stack is empty at the end of a grammatically correct sentence. The difference is that there are three push actions until the predictor can finally predict a word. Prediction is always based on the word that is stored on the top of the stack, and after three unsuccessful predictions on the first three words, these words are stored in the three layers of the stack, with the third word being on the top. The fourth word is predictable from the third word, which means that the push-pop neurons will perform a pop action on the stack, after which the second word will be on the top. The fifth word can be predicted from the second word, which means another pop action, and finally, the sixth word is predicted from the first one. As in the case of tail-recursion, the stack is empty after the presentation of a grammatically correct sentence.

It can be seen that in the case of a sentence with tail-recursion, only the top layer of the stack is used, whereas for parsing a six-word-long sentence with CER, three layers are used. Generally, the number of stack layers required for parsing a centre-embedded structure is half of the number of words in the sentence. If there are fewer layers, the network will categorize sentences with CER as agrammatical. Note that tail-recursion can be parsed without push-pop neurons and stack if you use simple copying from the input layer to a one-layer memory instead of a push action and deletion of the memory instead of a pop action. The difference between animals that cannot parse CER and humans can be that the former lack the stack and the gating mechanism, without which only local dependencies between words can be parsed.

It can be argued that the stack architecture in this form cannot explain why deeper embeddings are harder to process for humans. With this solution two levels of embedding (6-word-long sentences) are processed perfectly, whereas three or more levels are impossible. However, if we realize that every neural computation is prone to errors, we will see that the stack architecture with many layers also shows graceful degradation in performance as the level of embedding increases. If every push or pop operation in the stack has a small probability to result in imperfect transmission of information from one layer to another, then the more embedding the sentence have the more probable is its faulty parsing.

Training

While the architecture of the network described above is hand-crafted, its synaptic weights develop during training. For the training we randomly chose a subset (the learning set) from the grammatical sentences of either type of recursion. Training consisted of presenting the learning set several times and modifying the weights of the network. Testing was performed on the rest of the grammatical sentences that the network has not encountered before randomly mixed with agrammatical sentences. During testing no weight change occurred. The performance of the network was measured by its predictions for the following words in grammatical sentences during training and testing and its decisions on the grammaticality of the sentences at the end of the sentences. Note, that theoretically the maximum performance for prediction is 50% in grammatical sentences (obviously it was not measured for agrammatical sentences). Grammaticality judgement during testing measures if the network can differentiate grammatical from agrammatical sentences, while during training it is not very informative, since there were only grammatical sentences in that phase.

Different learning rules were used to modify the weights of the network. For the weights between the top layer of the stack and the predictor layer, a simple Hebbian learning rule was used. After the predictor layer tried to predict the next word and the push-pop neurons compared the prediction with the next word on the input, the input was copied to the predictor. Then learning occurred in this time step by increasing the synaptic weights between those neurons that were activated:

if $N_t=1$ *and* $N_p=1$ *then* $W_{tp}=W_{tp} + r$,

where N_t is a neuron on the top of the stack, N_p is a neuron on the predictor layer, W_{tp} is the synaptic weight between them and r is the learning rate (r was set to 0.01).

For modifying the synaptic weights of the push-pop neurons coming from the predictor and the input layer and the synaptic weights of the decision neuron coming from the top of the stack, the perceptron learning rule was used with threshold transfer function (Dayan and Abbott, 2005). This learning rule modifies the weights and the threshold to reach an output that is closer to a precalculated desired output. For this only local information is used: the activation of the input and the output layer (e.g., in the case of the decision neuron the input is the top of the stack) and the synaptic weights:

$W = W + h*(O – O') * I$ and $T = T – h*(O – O')$,

where W is the weight matrix between the input and the output layer, I is the input, O is the desired output, O' is the actual output, T is the threshold for the transfer function and h is the learning rate (h was set to 0.1). The desired output for the push-pop neurons is 1 (pop) if the prediction was correct (i.e., if the input layer and the predictor layer has the same activation pattern) and -1 (push) if the prediction was incorrect. For the decision neuron, the desired output is 0 if the top of the stack is empty and 1 otherwise. (Note that starting with nonzero random weights, the decision neuron automatically works well without learning.)

Weight modification occurred online, i.e. after the presentation of each word in the case of both learning rules. (We tried batch learning too, where weight modification occurs after the presentation of the whole training set. The network basically reached the same performance, however, we find it less realistic, and hence we used online learning for generating the figures in this paper.)

Those weights that copy activation from one layer to another were not trained, but set to the desired values from the beginning: between the input and the predictor, between the input and the top of the stack, and the weights between the layers of the stack. It might seem to be quite artificial that the weights of the stack are not trained but are precalculated. However, since it is a very simple structure it would be easy to train in an extended version of this model, just as the other copy weights.

Results and conclusions

Learning performance

Figure 3 a) and b) show the performance of the model during several training sessions and a test session averaged over 10 runs in the case of tail-recursion and CER, respectively. Performance is measured by the correctness of grammaticality judgement at the end of sentences (Decision) and by the correctness of the prediction for words during sentences (Prediction). Black data points represent performance during training while the last white data points represent performance during testing. Training was performed on a randomly chosen subset of the 336 grammatical sentences, while testing was performed on the rest of the grammatical sentences mixed with agrammatical sentences. Note, that the theoretical maximum for prediction performance is 50% in the case of grammatical sentences for both grammars (it was not measured for agrammatical sentences).

In the case of tail-recursion a training set composed of 10 grammatical sentences was usually enough for the network to generalize and reach perfect or almost perfect performance on novel sentences. For this about 5 training sessions were needed. In the case of CER, 30 training sentences presented for 11-12 training sessions were needed to reach the same performance. For both grammars, perfect performance on novel sentences is possible provided that every word-pair is presented during the training sessions. There is no generalization on the level of word-pairs; it is simply not possible since words are paired randomly. However, the network successfully generalizes on the level of sentences as can be seen from its performance on novel test sentences.

The network can also be trained with a mixed set of sentences conforming to tail-recursion and CER. With 30 sentences, only about 6-7 training sessions are needed to reach perfect performance which indicates faster learning than with CER sentences only. This is quite intuitive: tail-recursion seems easier to learn than CER since words forming a word-pair are presented immediately after each other. For the successful parsing of the grammars both memorizing the word-pairs and recognizing the particular structure is necessary. Since finding the words that depend on each other seems to be easier in the case of tail-recursion, we predict that it would help humans to learn CER if sentences were mixed with tail-recursive sentences.

Extensions

Since the performance of the network is based on the successful learning of word-pairs coupled with the push-pop operations of the stack, it can learn any language that is based on the balanced pairing of words in sentences. One example is the Dyck language of balanced parenthesis, which can be thought of as a mix of tail-recursion and centre-embedded recursion. E.g.: strings like *aaabbaabbabb* can be parsed by the model after learning that *a* and *b* are pairs. Another example is the palindrome (mirror) language; sentences like *abccba* can also be parsed by the network. The difference with centre-embedded recursion is that in this case word-pairs consist of two identical words.

The network cannot learn counting recursion in this form, since it has no module that would learn to categorize words to group A and group B. However, if we inserted a module that was able to categorize words on the predictor and on the input layer, it would make it possible to parse sentences with counting recursion too.

Conclusions

The main features of the neural network implemented here is the neurally implemented stack operated by gating neurons[1]. While it is well known that recursive sentences can be parsed by a symbolic stack, to our knowledge, there was no simple neural implementation of this structure until now. Symbolic models that could not be neurally implemented can be ruled out for being implausible (Christiansen and Chater, 2001b). What we show here is that the stack indeed can be neurally implemented, and it is quite simple provided that the push-pop operations are guided by gating connections.

We believe that gating will be found crucial for hierarchical tasks, just as for complex cognition in general (Gisiger et al., 2005; O'Reilly, 2006). The fact that it has readily evolved in a reinforcement-learning task in a simulated honeybee neural network (Soltoggio et al.,

[1] The stack follows a design borrowed from the chemical literature (Hjelmfelt et al., 1992) that rests on gating.

2007) supports this idea. We suggest that the introduction of the neural stack memory (push-down automaton) will also turn out to be substantial for any biological ‘hierarchical processor’. This is not to say that it is just a neural stack that is crucial for language, neither do we suggest that the stack architecture proposed here exists in a clean, isolated form in the brain, but it is likely that similar networks are embedded in the wider, language-related network context.

The performance of our network naturally depends on the depth of the stack, and as such it can be replaced by a finite-state automaton (Hopcroft and Ullman, 1979). However, in this sense human parsing ability is also limited: no person can parse sentences with arbitrarily many levels of embeddings (Pinker, 1994). The likely hierarchical processor (maybe even supramodal) in humans with normal development is Broca’s area (Friederici, 2006; Tettamanti and Weniger, 2006). Sadly, we know next to nothing about the relevant ‘internal wiring’ of this area: we propose that it is likely to contain a neural stack, wherein gating will be found important.

It would be premature to contemplate about the origin of stack-like neuronal systems in evolution and development. However, there seem to be two scenarios: either stacks are hard-wired (genetically coded) in our brain or we are born without them and the plasticity of our brain (under genetic control) makes us ‘ready’ to organize stacks during development. We think that the second scenario is more plausible but future work is needed to resolve these issues.

Acknowledgements

Thanks for the useful suggestions of Máté Lengyel, Chrisantha T. Fernando, Ferenc Huszár and Zoltán Szatmáry. This work was supported by the European Union Framework 6 (Grant IST-FET 1940). Support by the National Office for Research and Technology (NAP2005/KCKHA005) and by the National Scientific Research Fund (OTKA73047) is gratefully acknowledged.

References

Bickerton, D., 1990. Language and Species. Univ. Of Chicago Press.

Bodén, M., and Wiles, J., 2000. Context-free and context-sensitive dynamics in recurrent neural networks. Connection science 12, 197–210.

Chen, C.H., and Honavar, V., 1999. A neural-network architecture for syntax analysis. IEEE Trans Neural Netw 10, 94-114.

Chomsky, N., 1957. Syntactic Structures Mouton, The Hague.

Christiansen, M.H., and Chater, N., 1999. Toward a connectionist model of recursion in human linguistic performance. Cognitive Science: A Multidisciplinary Journal 23, 157 - 205.

Christiansen, M.H., and Chater, N., 2001a. Connectionist psycholinguistics: capturing the empirical data. Trends in Cognitive Sciences 5, 82-88.

Christiansen, M.H., and Chater, N., Connectionist psycholinguistics in perspective, in: Christiansen, M. H. and Chater, N., Eds.), Connectionist Psycholinguistics. Greenwood Publishing Group 2001b, pp. 19-76.

Corballis, M.C., 2007a. Recursion, language, and starlings. Cognitive Science 31, 697-704.

Corballis, M.C., 2007b. On phrase structure and brain responses: a comment on Bahlmann, Gunter, and Friederici (2006). J Cogn Neurosci 19, 1581-3.

Dayan, P., and Abbott, L.F., 2005. Theoretical neuroscience: computational and mathematical modeling of neural systems. The MIT Press, Cambridge, MA.

De Vries, M.H., Monaghan, P., Knecht, S., and P., Z., 2008. Syntactic structure and artificial grammar learning: The learnability of embedded hierarchical structures. Cognition doi:10.1016/j.cognition.2007.09.002.

Deacon, T.W., Multilevel selection in a complex adaptive system: the problem of language origins, in: Weber, B. H. and Depew, D. J., Eds.), Evolution and learning: The Baldwin effect reconsidered, MIT Press, Cambridge, MA 2003, pp. 81-106.

Eens, M. (Ed.), 1997. Understanding the complex song of the European starling: An integrated ethological approach. Academic Press, New York.

Elman, J.L., 1991. Distributed Representations, Simple Recurrent Networks, And Grammatical Structure. Machine Learning 07, 195-225.

Fedor, A., Ittzés, P., and Eörs, S., The biological background of syntax evolution, in: Bickerton, D. and Szathmáry, E., Eds.), Biological foundations and origin of syntax, Vol. 3. MIT Press, Cambridge, MA 2009, pp. 299-324.

Fitch, W.T., and Hauser, M.D., 2004. Computational constraints on syntactic processing in a nonhuman primate. Science 303, 377-80.

Friederici, A.D., 2006. Broca's area and the ventral premotor cortex in language: functional differentiation and specificity. Cortex 42, 472-5.

Gentner, T.Q., Fenn, K.M., Margoliash, D., and Nusbaum, H.C., 2006. Recursive syntactic pattern learning by songbirds. Nature 440, 1204-7.

Gisiger, T., Kerszberg, M., and Changeux, J.P., 2005. Acquisition and performance of delayed-response tasks: a neural network model. Cereb Cortex 15, 489-506.

Hauser, M.D., Chomsky, N., and Fitch, W.T., 2002. The faculty of language: what is it, who has it, and how did it evolve? Science 298, 1569-79.

Hjelmfelt, A., Weinberger, E.D., and Ross, J., 1991. Chemical implementation of neural networks and Turing machines. Proc Natl Acad Sci U S A 88, 10983-7.

Hopcroft, J.E., and Ullman, J.D., 1979. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, MA.

Hurford, J., 2007. The Origins of Meaning. Oxford Univ. Press, Oxford.

Katz, P.S., 2003. Synaptic Gating: The Potential to Open Closed Doors. Current biology : CB 13, R554-R556.

Maynard Smith, J., and Szathmáry, E., 1995. The Major Transitions in Evolution. Freeman, Oxford.

Mazzoni, P., Andersen, R.A., and Jordan, M.I., 1991. A more biologically plausible learning rule for neural networks. Proceedings of the National Academy of Sciences of the United States of America 88, 4433-4437.

O'Reilly, R.C., 1996. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. Neural Comput. 8, 895-938.

O'Reilly, R.C., 2006. Biologically based computational models of high-level cognition. Science 314, 91-4.

Perruchet, P., and Rey, A., 2005. Does the Mastery of Center-Embedded Linguistic Structures Distinguish Humans from Nonhuman Primates? Psychonomic Bulletin & Review 12, 307-313.

Pinker, S., 1994. The Language Instinct. Penguin.

Robinson, J.G., 1984. Syntactic Structures in the Vocalizations of Wedge-Capped Capuchin Monkeys, Cebus Olivaceus. Behaviour 90, 46-78.

Rodriguez, J., Wiles, J., and Elman, J.L., 1999. A recurrent neural network that learns to count. Connection Science 11, 5-40.

Soltoggio, A., Dürr, P., Mattiussi, C., and Floreano, D., Evolving neuromodulatory topologies for reinforcement-like problems, IEEE Congress on Evolutionary Computation, IEEE Press 2007, pp. 2471-2478.

Sun, G.Z., Giles, C.L., Chen, H.H., and Lee, Y.C., The Neural Network Pushdown Automaton: Architecture, Dynamics and Training., in: Giles, C. L. and Gori, M., Eds.), Adaptive Processing of Sequences and Data Structures, Springer 1998, pp. 296-345.

Szathmáry, E., Towards an understanding of language origins, in: Barbieri, M., (Ed.), Codes of Life, Springer-Verlag 2007, pp. 283-313.

Tettamanti, M., and Weniger, D., 2006. Broca's area: a supramodal hierarchical processor? Cortex 42, 491-4.

Wiles, J., Watson, J., Tonkes, B., and Deacon, T., 2005. Transient Phenomena in Learning and Evolution: Genetic Assimilation and Genetic Redistribution. Artif. Life 11, 177-188.

Zuberbühler, K., 2002. A syntactic rule in forest monkey communication. Animal Behaviour 63, 293-299.

Figure captions

Fig. 1. Tail-recursion (a), counting recursion (b) and centre-embedded recursion (c). A and B represent word categories and As and Bs with the same index form word-pairs. Word-pairs imply only local dependencies in a) but also long-range dependencies in c). There are no word-pairs in b) (Corballis, 2007a; Corballis, 2007b).

Fig. 2. Architecture of the proposed neural network model. The input layer receives words from sentences one-by-one. The stack is represented with three layers with bidirectional inter-layer connections. The predictor layer tries to predict the next word based on the word that is stored at the top of the stack (stack layer 1). There are two push-pop neurons (P) that has gating connections (dashed lines) on the inter-layer connections of the stack. Inhibitory gating connections are marked by a circle at the end and excitatory gating connections are marked by a diamond at the end. The decision neuron (D) gives grammaticality judgement on the sentence. Copying connections that are not trained are indicated by empty arrows. Synapses between the top of the stack and the predictor are indicated by a thick arrow and trained by the Hebbian learning rule. All other synapses are trained by the perceptron learning rule.

Fig. 3. Performance of the neural network model on a) tail-recursion and b) centre-embedded recursion averaged over 10 runs. Performance is measured by the correctness of grammaticality judgement at the end of sentences (Decision) and by the correctness of the prediction for words during sentences (Prediction). Black data points represent performance during training while the last white data points represent performance during testing. Training was performed on 10 and 30 randomly chosen grammatical sentences in the case of tail-recursion and centre-embedded recursion, respectively. Testing was performed on the rest of the grammatical sentences mixed with agrammatical sentences.

Fig. 1a

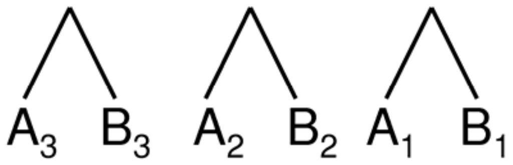


Fig. 1b

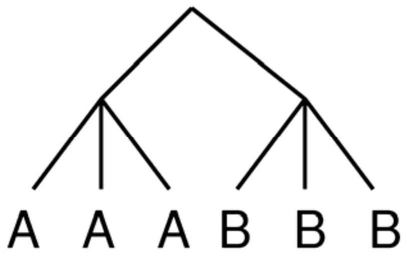


Fig. 1c

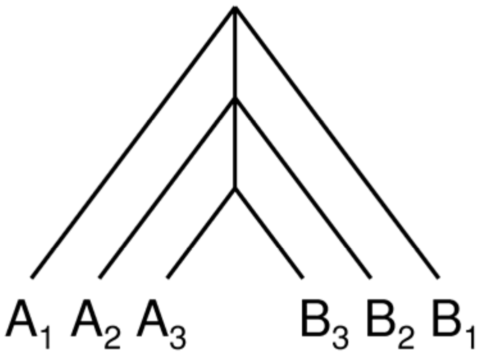


Fig.
2

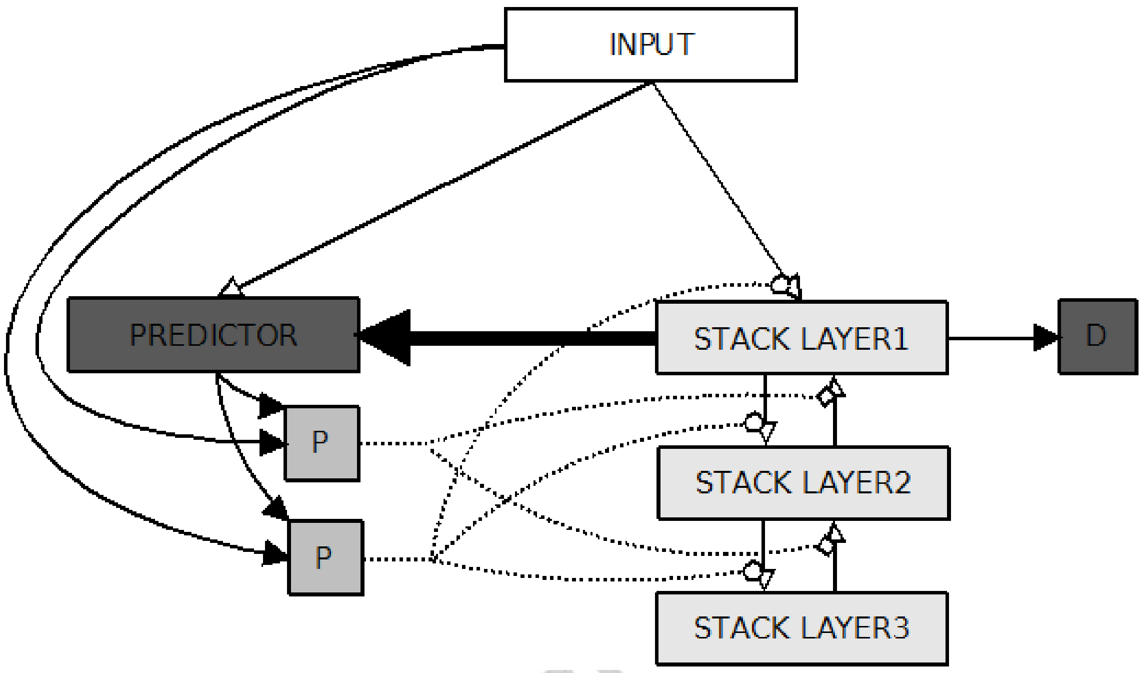
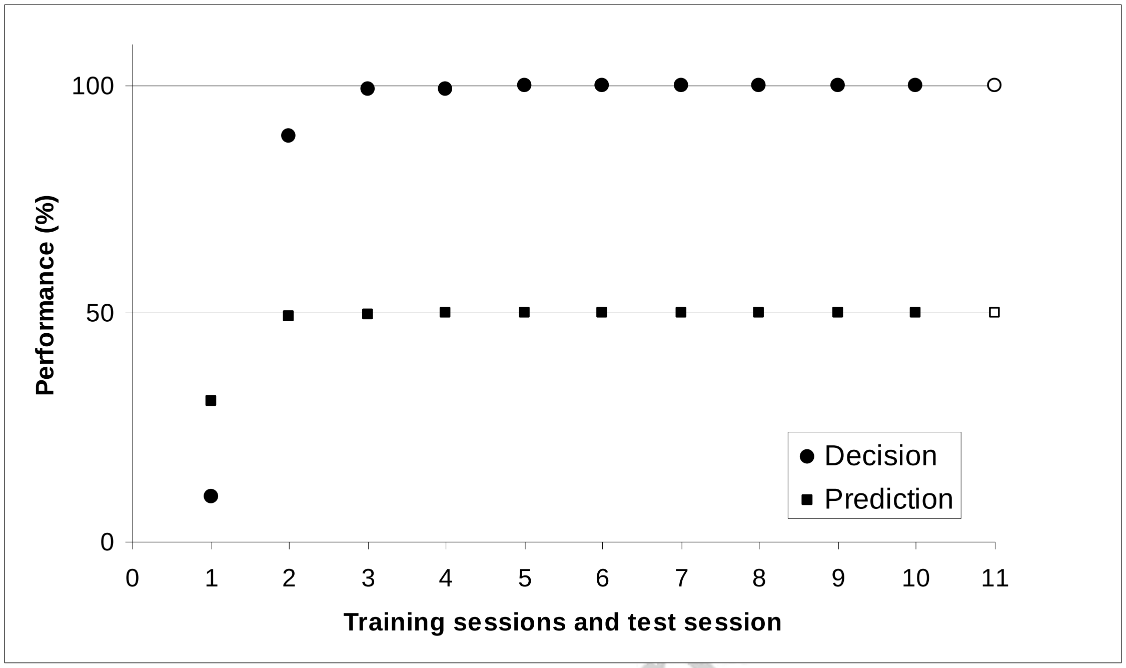
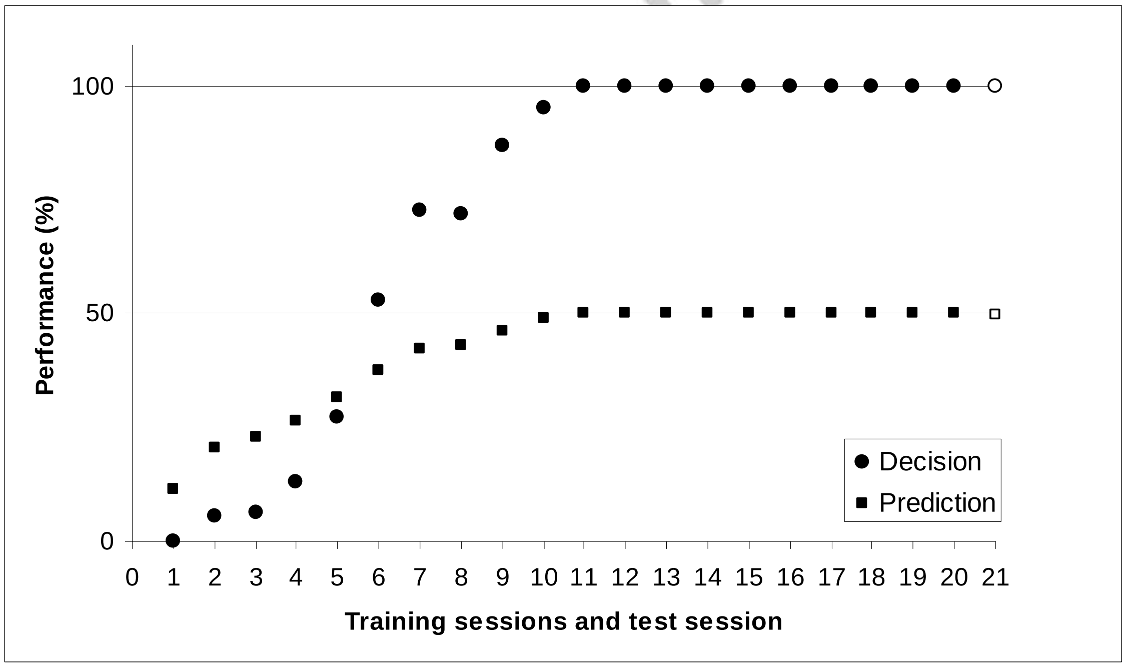

INPUT
PREDICTOR
STACK LAYER1
D
P
STACK LAYER2
P
STACK LAYER3

Fig. 3



100
50
0
Performance (%)
0 1 2 3 4 5 6 7 8 9 10 11
Training sessions and test session
● Decision
■ Prediction



100
50
0
Performance (%)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Training sessions and test session
● Decision
■ Prediction