The hierarchical prediction network: towards a neural theory of grammar acquisition

Gideon Borensztajn (gborensztajn@uva.nl)
Willem Zuidema (zuidema@uva.nl)
Rens Bod (rens.bod@uva.nl)
All authors: Institute of Logic, Language and Computation, 904 Science Park
1098 XH, Amsterdam, The Netherlands

Abstract

We present a biologically inspired computational framework for language processing and grammar acquisition, called the hierarchical prediction network (HPN). HPN fits in the tradition of connectionist models, but it extends their power by allowing for a substitution operation between the nodes of the network. This, and its hierarchical architecture, enable HPN to function as a full syntactic parser, able to emulate context free grammars without necessarily employing a discrete notion of categories. Rather, HPN maintains a graded and topological representation of categories, which can be incrementally learned in an unsupervised manner. We argue that the formation of topologies, that occurs in the learning process of HPN, offers a neurally plausible explanation for the categorization and abstraction process in general. We apply HPN to the task of semi-supervised 'grammar induction' from bracketed sentences, and demonstrate how a topological arrangement of lexical and phrasal category representations successfully emerges.

Introduction

Our understanding of the world around us is organized in categories. There is nothing more basic than categorization to our thought, reasoning, perception, action, learning and speech. Categories are necessary for generalization from concrete observations to general rules that apply to novel situations, and to productively use language.

Of course, categories also play a central role in linguistic theory. The sets of rules that describe grammars are essentially systems of relations between categories. Yet, within formal linguistics no convincing account exists of how syntactic categories are acquired. Within the tradition of generative grammar there has been little interest in this question, as humans are believed to be born equipped with a full fletched language faculty (a universal grammar) with abstract adult syntactic categories already in place. A major obstacle for explaining the incremental acquisition of categories is their status as set theoretical entities, where category membership is either true or false, implying that a category exists before it has any members.

Cognitive linguistics, in contrast, proposes that syntactic categories are prototype-based, and that category membership is graded (Lakoff, 1987). The acquisition of prototypical categories can be modelled using a localist connectionist approach, such as exemplified by the Kohonen self-organizing map (Kohonen, 1995).

One often cited advantage of Kohonen networks is that they account for the development of topologies, such as have been demonstrated not just in the visual system (e.g.. the orientation columns of complex cells, Hubel & Wiesel, 1968), but everywhere in the brain. Why would the brain invest such an effort in developing topologies? The most plausible answer, we find, is that the topology in the brain serves to encode graded category membership. The brain infers the category of a neural assembly from its topological position on the cortical surface, whether it is for seeing, walking or talking.

Another inspiration for our work comes from theories of cortical information processing that stress the hierarchical and columnar organization in the (visual) neocortex, such as the fragment-based hierarchy approach (Ullman, 2007) and the Memory Prediction Framework (MPF) (Hawkins & Blakeslee, 2004). The key elements of the latter framework are that (i) categories in the neocortex encode *temporal sequences* (of patterns), (ii) the neocortex stores categories in an *hierarchical* fashion, (iii) as one goes up in the hierarchy, categories are formed that are progressively more *invariant* and more *temporally compressed*, (iv) the main function of the cortex is *prediction* of future events, and this is achieved by 'unfolding' the temporally compressed categories to the lower levels.

In the current paper these ideas are generalized for language, and integrated in a self-organizing network. The resulting model, besides accounting for the gradedness of categories, also incorporates the notion of phrase structure.

A neural theory of syntax

Starting point of the research in this paper is the assumption that there exists a uniform cortical mechanism that underlies categorization and processing within different modalities. The analogy between visual and linguistic processing leads to a proposal for a neural theory of language processing and acquisition, and a computational implementation thereof, which we call the Hierarchical Prediction Network (HPN). The theory is founded on the following hypotheses about the neural representation of a grammar:

- There exist cell assemblies in the language area of the cortex that function as neural correlates of graded syntactic categories[1].
- These 'syntactic' cell assemblies represent *temporally compressed* word sequences (phrases) which can 'unfold' to *predict* words or assemblies in a lower level.
- The topological and hierarchical arrangement of 'syntactic' assemblies in the cortical hierarchy constitutes a grammar.

[1] We do not actually adhere to a strict separation between syntax and semantics, but in this work we focus on syntax alone.

Still, there is one more important aspect of human language that our model has to deal with. It is often claimed that the only thing special about human language is recursion (Hauser, Chomsky, & Fitch, 2002). Recursion happens to be a major stumbling block for most connectionist models of language. We argue, that all that is needed to explain the phenomenon of recursion from a cognitive point of view is a basic ability for neural systems to perform *substitution*, and that recursion is an epiphenomenon. It is clear what substitution means in a symbolic framework, because it is implicitly assumed that variables are global and their content can be replaced. Within a biological context however, which lacks the notion of variables, it is not so obvious what substitution is, nor is it trivial, as with the related problem of binding, how it can be instantiated in a connectionist model for language processing (Fodor & Pylyshyn, 1988).

In this paper we propose a neural network solution for substitution that renders HPN potentially much more powerful than traditional connectionist networks. The price is that we have to postulate a capacity for the nodes in our network to transmit and store richer information than just activation levels, as will be detailed in the next section.

Whereas in a symbolic framework substitution relations between category members are typically assumed to be be innate, in a neurally motivated model of grammar and category learning the acquisition of substitutability relations (between nodes in the network) becomes the central question. We hypothesize that a graded measure of substitutability is realized as a distance between neural assemblies within a meaningful topology. The central claim of this work is thus that language acquisition is in essence equivalent to learning a network topology.

HPN architecture

The hierarchical prediction network (HPN) consists of an *input layer* composed of *input nodes* on top of which there are one or more *compressor layers* with so-called *compressor nodes*. The nodes have a fixed position in the network, but they also develop representations in a virtual space, which we call the *substitution space*. The internal representation is learned through the interactions that take place in the network, and will eventually reflect the distribution of the input data.

Input nodes interact with the external environment; in the language domain every input node is assumed to correspond to a unique word from the lexicon, and it fires whenever this word is presented within a sentence.

The *compressor nodes* correspond to sequences of words (phrases). The name *compressor nodes* implies that they temporally compress a sequences of nodes from the lower layers (as was hypothesized by the MPF). Compressor nodes have two or more ordered *slots*, with which they form a fixed unit.

An ordered set of slots on a compressor node constitutes a *production*. A production is executed by matching each one of the slots in a fixed order to an input or compressor node through bottom-up activation of the slot. An HPN production thus resembles a rewrite rule with the exception that its slots are not associated with any particular non-terminal. When all of its slots in turn have received bottom-up activation, the production is *completed*, and the compressor node fires.

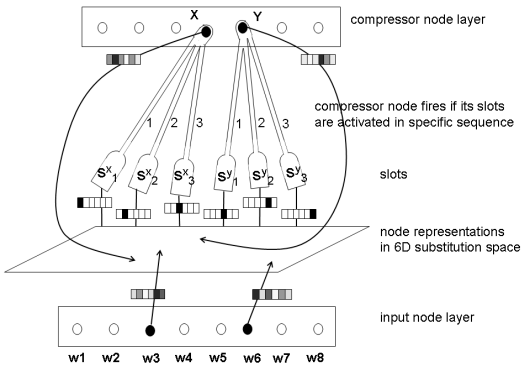

Figure 1: HPN architecture.

In HPN, the slots fullfill the role of *physical* substitution sites where nodes are coupled to each other. Such a local coupling mechanism serves as a neural correlate for substitution in formal grammars, where substitution is defined globally. Every slot in HPN offers an independent substitution site in the context of which a set of nodes are more or less substitutable. Thus, the slots form a basis for a *substitution space*, with respect to which the internal representations of both the input and compressor nodes in HPN are defined. Our implementation circumvents the need for full connectivity, and at the same time enables node substitution: when a node fires, its internal representation is serially transmitted over a single link to a central switch board (substitution space), where it is transferred to the slots that best fit the node's representation. This is illustrated in Figure 1.

Substitutablity between two nodes is given as some distance measure on the metric of substitution space. Thus, regions in substitution space define a continuum of categories, and a node's representation in substitution space defines its graded membership to one or more categories.

Another feature that distinguishes HPN from conventional neural networks is the fact that HPN keeps track of (serial) activation paths through the network as an input sequence is processed. Once (and provided) the neural network has access to the activation path, it can benefit from information that is relevant for the internal interpretation of the sequence with respect to its components, such as feature grouping (for visual processing) and phrase structure. These are tasks that conventional neural networks are not good at.

Typically, many processes run in parallel in the network, and each is associated with a distinctive path. Different paths compete with each other for the best internal interpretation of the sequence. In order to differentiate between alternative paths (possibly crossing the same node multiple times), HPN

keeps so-called 'path connectors' in the slots of every visited compressor node along the activation path. The path connectors store a pointer to the node that has bound to the slot (the 'sender node'), together with the 'time of activation', which equals the processed part of the input sequence (the current position). The path connectors allow to distinguish different *active states* of a single HPN node, much like the states in an Earley chart parser (Earley, 1970). The information needed to update the state of the nodes (current sentence position and sender node identity) is transmitted along the activation path, through the links mentioned before.

The solution for the binding problem in HPN can be ascribed to the ability of compressor nodes to temporarily store (multiple) path connectors. By virtue of locally stored pointers from slots to 'bound' nodes, HPN is able in principle to represent parse trees of unlimited depth. An ordered sequence of such pointers corresponds to a *stack* in symbolic parsers, since it constitutes a distributed memory of the productions and input nodes involved in the parse.

HPN as a syntactic parser

Parsing in HPN takes the form of an interaction between *parallel bottom-up* activation of input and/or compressor nodes and *serial top-down* execution of productions, as ordered sequences of predictions. The parse starts with bottom-up activation of productions whose left slot matches the meaning representation of an activated input node. Slots must be matched in the correct order, and adjoining slots within a single production accept only adjoining portions of the sentence. (This is why we need to keep track of the 'time of activation' in the slots). When all slots of a production are matched, the compressor node fires, and transmits its internal representation, such that it in turn can be matched with a slot of another production. Figure 2 shows some typical parsing scenario's.

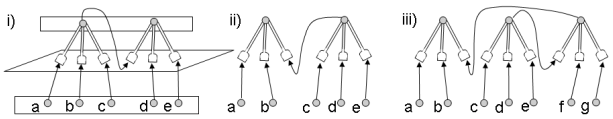

Figure 2: Scenario's of three HPN parses. i) ((a b c) d e), ii) (a b (c d e)), iii) (a b ((c d e) f g)).

Although it is possible to implement a parallel version of the HPN parser, we have implemented a serial version of HPN that roughly parallels left corner parsing, and uses backtracking to search the space of possible parses. Left corner parsing is a cognitively quite plausible incremental parsing strategy that combines top-down and bottom-up parsing. For details, see (Rosenkrantz & Lewis II, 1970).

A *parse* in HPN is a trajectory through the nodes of the physical network that binds a set of productions together through path connectors. A parse is called successful if

1. every successive input word in the sentence is predicted by one of the productions involved, in the correct order.
2. every slot of every compressor node in the parse is bound either to an input node (a word), or to the root of another compressor node (a phrase); there is only a single compressor node with an unbound root.
3. adjoining slots within a production can only be bound to nodes that processed adjoining portions of the sentence.

A *derivation* is an ordered sequence of time indexed productions (states) and bindings, that fully determines the followed trajectory through the network space upon a successful parse. As an example we give a derivation for the sentence *Sue eats a sandwich* in Table 1.

$X1$	$\rightarrow$	1 2	1	$>=<$	Sue
$X2$	$\rightarrow$	3 4	3	$>=<$	$eats$
$X3$	$\rightarrow$	5 6	5	$>=<$	a
2	$>=<$	$X2$	6	$>=<$	$sandwich$
4	$>=<$	$X3$			

Table 1: A derivation of the sentence *Sue eats a sandwich*. Capital letters indicate compressor nodes, $italics$ indicate input nodes, numbers indicate slots, $>=<$ indicates a binding.

Representation of context free grammars

Context-free grammars (CFG) are a special case of an HPN grammar with a single compressor layer, and they can be represented as a set of input nodes and compressor nodes with appropriate meaning representations. We show this by sketching a conversion procedure from a CFG grammar to an HPN representation, such that those and only those sentences that are successfully parsed by the CFG grammar are successfully parsed by the HPN grammar. For convenience, we will represent the nodes and slots as vectors with respect to a basis of slots. We can then compute the match between a firing node and a slot as the inner product between their representations (if the inner product equals 0 then there is no match).

S	$\rightarrow$	NP VP (1.0)
NP	$\rightarrow$	PropN (0.2) ‖ N (0.5) ‖ N RC (0.3)
VP	$\rightarrow$	VI (0.4) ‖ VT NP (0.6)
RC	$\rightarrow$	WHO NP VT (0.1) ‖ WHO VP (0.9)
VI	$\rightarrow$	walks (0.5) ‖ lives (0.5)
VT	$\rightarrow$	chases (0.8) ‖ feeds (0.2)
N	$\rightarrow$	boy (0.6) ‖ girl (0.4)
PropN	$\rightarrow$	John (0.5) ‖ Mary (0.5)
WHO	$\rightarrow$	who (1.0)

Table 2: Example probabilistic context-free grammar. Probabilities are indicated in brackets.

1. Create a separate HPN production for every non-unary rule expansion, and assign unique and orthogonal representations to its slots. (For example, $S \rightarrow (1000000000)\ (0100000000)$). The representation of the compressor nodes will be determined later.
2. For every non-unary production in the CFG, change the representations of all non-terminals occurring on its right

hand side by adding the slot vectors (as assigned in step 1) with which they are associated (using vector addition);
3. For every unary production in the CFG, copy or add the representation of the non-terminal on the left hand side to the representation of the non-terminal or terminal on the right hand side (do this recursively).
4. Assign the appropriate non-terminal representation to the roots of HPN productions, and create input nodes using the representations computed in step 3. Discard all unary productions, and unused non-terminals.

The conversion procedure is illustrated in Figure 3 (using informal notation, and with unary productions added for clarity), for the CFG grammar with recursive relative clauses shown in Table 2.

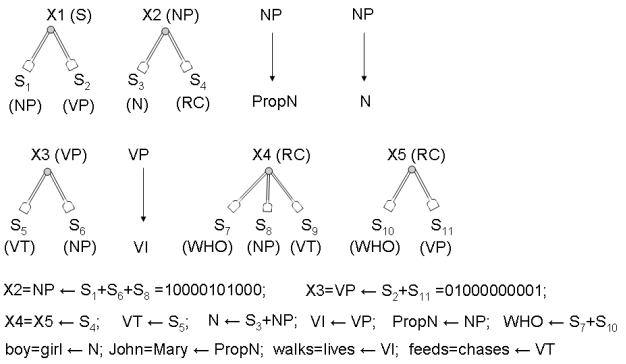

Figure 3: Conversion procedure from CFG to HPN. The slot indices reflect their only non-zero component.

One can easily check that given these representations there is only one way for HPN to parse for example the sentence *boy who lives chases Mary*.

It is easy to modify the conversion procedure such that it converts a probabilistic context free grammar (PCFG) into a probabilistic HPN. To do so, during the construction of node representations (step 2 and 3) one must multiply the representation of the left hand side by the probability of their respective expansion in the PCFG. The inner product between a node and a slot representation now gives the probability of their binding, and the product of the probabilities of all bindings involved in an HPN derivation gives the HPN parse probability. Table 3 gives the 'probabilistic' node representations (assuming the HPN productions of Figure 3). It can be shown that, using this conversion procedure, for any sentence generated by the example artificial grammar, every PCFG parse has an HPN counterpart with the same phrase-structure and the same probability.

Learning

While in the previous section we have shown that a syntax can be *represented* in HPN, the added value of the connectionist approach is in its ability to actually *learn* syntactic representations. Within the symbolic paradigm learning syntactic categories from context is a discrete process: a familiar strategy is to *merge* two words into a single category when they appear in similar contexts (Stolcke & Omohundro, 1994). One of the advantages of defining category membership by means of a topology is that it allows category meanings to change continuously while learning.

Compressor nodes	
X2 (NP)	(.3 0 0 0 0 .3 0 .3 0 0 0)
X3 (VP)	(0 .6 0 0 0 0 0 0 0 0 .6)
X4 (RC)	(0 0 0 .1 0 0 0 0 0 0 0)
X5 (RC)	(0 0 0 .9 0 0 0 0 0 0 0)
Input nodes	
John = Mary = (.1 0 0 0 0 .1 0 .1 0 0 0)	
lives = walks = (0 0 0 0 .5 0 0 0 .5 0 0)	
boy = (.3 0 0 .6 0 .3 0 .3 0 0 0); girl = (.2 0 0 .4 0 .2 0 .2 0 0 0)	
chases = (0 0 0 0 .8 0 0 0 .8 0 0); feeds = (0 0 0 0 .2 0 0 0 .2 0 0)	
who = (0 0 0 0 0 0 1 0 0 1 0)	

Table 3: Node representations for probabilistic HPN.

In order to induce a meaningful topology, one must somehow induce the meaning representations from the distribution implicit in the corpus. HPN uses a similar strategy as above to identify meaningful relations between two words, but it makes the words more substitutable in a gradual way, rather than discretely, by decreasing their distance in substitution space upon encountering similar contexts.

In HPN parsing and learning are complementary, as node representations are dynamically adapted after every parse. Following is a sketch of the algorithm:

1. Initialization. Create input nodes with random representations for every distinct word in the corpus. Specify the number of productions of each size (i.e., number of slots), and create a compressor node with random representation for each production. Initialize all slot representations orthogonally to each other.
2. Parsing. For every sentence, let HPN compute the most probable parse (as explained above). Recover the productions and bindings involved, using the path connectors.
3. Learning. For every binding adjust the representation of 'winning node' $\mathbf{n}$ that participated in slot $\mathbf{s}$ according to $\Delta\mathbf{n} = \lambda * \mathbf{s}$, with (decreasing) learning rate λ. Also, adjust the representations of the nodes in the neigborhood h of the 'winning nodes', in proportion to their distance in substitution space.

As can be inferred from the learning step, the internal representations of words and syntactic categories are gradually changed from concrete (i.e., not correlated with other node representations) to abstract, when they participate in more slots. Two lexical nodes that often participate in the same slot(s) will be gradually merged into a single part of speech category. By shrinking the neighborhood, and decreasing λ with time, as in a Kohonen network, a topology over node representations is incrementally induced in substitution space, based on the corpus distribution.

Evaluation

The learning algorithm was evaluated on the artificial language of Table 2. We generated at random 1000 distinct sentences with until 5 levels of recursion from the CFG grammar, and split these in a training corpus of 800 sentences (with brackets included) and a test corpus of 200 sentences (without brackets). We initialized the HPN network by creating 10 productions with 2 slots, and 5 productions with 3 slots, and we set all compressor and input node representations to random initial values. The learning rate decreased from $\lambda = 0.3$ to $\lambda = 0.05$ and the neighborhood from $h = 10$ to $h = 0.01$. Figure 4 shows the representations of the input nodes after learning has completed (scaled to two dimensions using Matlab's `cmdscale`). Also shown are the average compressor node representations that map to the symbolic labels of the gold standard parses (NP, etc.). From the figure it can be seen

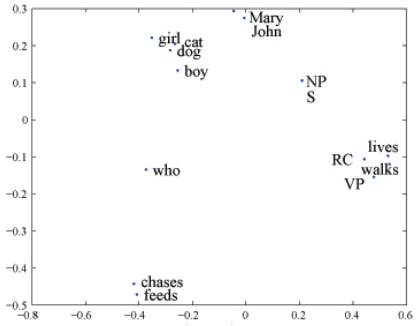

Figure 4: Substitution space with HPN node representations

that the nouns, proper nouns, intransitive and transitive verbs cluster into postag categories, and some of the higher order categories are mapped in substitution space between words with which they are intuitively substituted. To test whether the induced HPN grammar had indeed developed useful representations, we let it parse the test corpus with, and evaluated precision (UP) and recall (UR) of the found constituents. The resulting values of UP=UR=0,864 indicate that the induced HPN grammar approximates the original grammar to a fair degree.

Unlike connectionist models that are trained with error back-propagation, training with HPN is scalable to realistic size corpora, and can be done in a single pass through the data. This created the opportunity to evaluate HPN on the Eve corpus from the CHILDES database (MacWhinney, 2000). Figure 5 shows very preliminary results on 2000 consecutive child utterances from the second half of the Eve corpus, where we used the brackets available from the dependency annotation. The interesting clusters are accentuated. Note, that for this experiment an earlier version of the algorithm was used, that did not employ a neigborhood function.

Relation to other modelling work

Unsupervised induction of syntactic labels has been done in the 'symbolic' paradigm, with best results through *Bayesian*

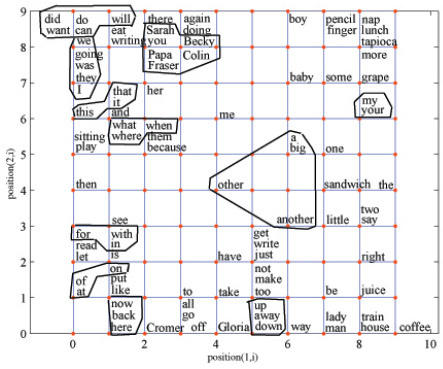

Figure 5: Representations of Eve's 100 most frequent words

Model Merging (Stolcke & Omohundro, 1994; Borensztajn & Zuidema, 2007). HPN shares the division of work between the syntagmatic and the paradigmatic components of the model (e.g., the chunk operator is replaced by compressor nodes). A drawback of all symbolic approaches to categorization, which is inherent to the discrete nature of the categories, is that they suffer from the lack of well-motivated criteria for preventing over-generalization, and for deciding on category boundaries. These problems are mitigated in HPN, with the assumption of a continuous substitution space.

Most connectionist work on sentence processing has been carried out in the tradition of Elman's Simple Recurrent Network (SRN) (Elman, 1991), using variations of fully distributed multi-layer perceptron networks and error backpropagation. Such networks do not employ a notion of categories at all, and consequently are not suited for capturing phrasal structure. Much attention has been paid to the abilities of the SRN to deal with 'quasi-recursion' (Tabor, 2000; Christiansen & Chater, 1999). Typically, a word prediction task is used, and a similar artificial grammar as ours. There is reason to believe, that because of its architecture HPN will perform well on this task with highly recursive sentences, but benchmark tests still remain to be done.

HPN and SRN take a very different approach in their treatment of time: the SRN learns temporal sequences by feeding the activation of the hidden layer at time t-1 back to a so-called context layer. As a consequence of the linear history, the internal (word) representations formed in the hidden layer of the SRN are approximately Markovian. In contrast HPN, inspired by (Hawkins & Blakeslee, 2004), uses hierarchical temporal compression, and therefore maintains a non-linear representation of a word's history. HPN constructs its word represention depending on an optimal phrasal decomposition of the sentence: the context of a word depends on the slot position that the word is bound to within a certain HPN production. Moreover, unlike SRN's, HPN maintains explicit representations of higher order syntactic constituents, which in turn constitute part of the context for the terminals.

Yet another class of connectionist models of syntactic processing emulates symbolic parser behavior by proposing

mechanisms for translating the concept of compositionality into a connectionist setting, such that compositional structures can be represented. Examples of these are Harmony Theory (HT) (Prince & Smolensky, 1997), and derivatives of RAAM (Pollack, 1990), such as SPEC (Miikkulainen, 1996). In our opinion, in brief, the problem with the proposed architectures is that, in contrast to HPN, they are not capable of autonomously finding structural (tree) representations, without relying on external or innate knowledge of the node meanings. HPN, rather than constructing compositional representations, deconstructs sentences into tree-like representations because it is confined to map their interpretations onto the hierarchical and topological structure of the network. As such, HPN offers a biologically motivated explanation for the origin of compositionality, that suggests that the superficially compositional structure of language (as well as vision) arises from the hierarchical processing that takes place in the brain.

Discussion and Conclusions

We have presented a novel connectionist network, the Hierarchical Prediction Network. A key innovation of HPN is its ability to temporarily *bind* the nodes in the network (by means of pointers), and thereby provide a neural substrate of substitution. This is a significant achievement, because it equips HPN with a mechanism for some kind of 'variable' manipulation, and with the ability to represent systematic relations over 'variables' (i.e., HPN exhibits systematicity). The lack of this ability has been a major source of critique against the suitability of classical connectionist models to deal with language processing (Fodor & Pylyshyn, 1988). The fact that the nodes of the network can engage in meaningful relations makes HPN much more powerful than conventional neural networks, since it transforms HPN into a structured knowledge base (i.e., a grammar) rather than a loose collection of nodes (e.g., see (Marcus, 2001)).

By virtue of a mechanistic explanation for substitution, HPN creates a synthesis between the traditional connectionist and symbolist frameworks. Yet, even though HPN borrows some notions from the parsing field, its implementation is fully connectionist: HPN's internal representations are induced from arbitrary initial representations, and all interactions are local. We believe that the main contribution of our work is therefore that it formulates *sufficient* conditions for a connectionist solution of the structure encoding problem. Major departures from the symbolic paradigm are that the classical notion of a category is replaced by that of a graded topology, within which neighboring nodes are substitutable, and that 'rules' (the HPN productions) have local rather than global scope.

HPN simulates the fact that categories emerge gradually, and their meaning grows incrementally with time from concrete to abstract, as the nodes associated with the category integrate in the network topology. Such a strategy makes HPN very well suited as a model for language acquisition, because it offers a plausible explanation for the tricky question of how syntactic categories can be bootstrapped from scratch.

Empirical research in child language acquisition seems largely to support such an approach to learning. In Tomasello's theory of Usage Based Grammar (UBG) (Tomasello, 2003) the gradual acquisition of syntactic categories is referred to as item-based learning: in the early developmental stages linguistic constructions are typically learned case-by-case, often around specific verbs, without reference to a general syntactic category. Only gradually children learn to generalize across constructions, when they acquire an adult-like grammar that uses system-wide syntactic categories. HPN seems to offer a promising direction for modeling grammar acquisition in this tradition.

Acknowledgements

We want to thank Stefan Frank and Remko Scha for valuable discussions and comments.

References

Borensztajn, G., & Zuidema, W. (2007). *Bayesian model merging for unsupervised constituent labeling and grammar induction* (Tech. Rep.). (ILLC Technical Report PP-2007-40)

Christiansen, M. H., & Chater, N. (1999). Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, *23*(2), 157-205.

Earley, J. (1970). An efficient context-free parsing algorithm. In *Communications of the acm* (Vol. 13, p. 94-102).

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *7*, 195-225.

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

Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The faculty of language: what is it, who has it, and how did it evolve? *Science*, *298*, 1569-1579.

Hawkins, J., & Blakeslee, S. (2004). *On intelligence*. New York: Henry Holt and Company.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey visual cortex. *Journal of Physiology*, 215-243.

Kohonen, T. (1995). *Self-organizing maps*. Berlin: Springer Verlag.

Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about the mind*. Chicago, IL: University of Chicago Press.

MacWhinney, B. (2000). *The childes project: Tools for analyzing talk. third edition*. Mahway, NJ: Lawrence Erlbaum Associates.

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

Miikkulainen, R. (1996). Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, *20*, 47-73.

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

Prince, A., & Smolensky, P. (1997). Optimality: from neural networks to universal grammar. *Science*, *275*(5306), 1604-10.

Rosenkrantz, D., & Lewis II, P. (1970). Deterministic left corner parsing. In *11th annual symposium on switching and automata theory* (p. 139-152). New York: IEEE Press.

Stolcke, A., & Omohundro, S. M. (1994). Inducing probabilistic grammars by bayesian model merging. In *Proceedings of the second international colloquium on grammatical inference and applications (icgi'94)* (Vol. 862, pp. 106–118). Berlin: Springer Verlag.

Tabor, W. (2000). Fractal encoding of context free grammars in connectionist networks. *Expert Systems*, *17*(1), 41-56.

Tomasello, M. (2003). *Constructing a language: A usage-based theory of language acquisition.* Cambridge, MA: Harvard University Press.

Ullman, S. (2007). Object recognition and segmentation by a fragment-based hierarchy. *Trends in Cognitive Science*, *11*(2), 58-64.