The neural basis of structure in language

Bridging the gap between symbolic and connectionist models of language processing

Gideon Borensztajn

The neural basis of structure in language

Bridging the gap between symbolic and connectionist models of language processing

ILLC Dissertation Series DS-2011-11

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 904
1098 XH Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The neural basis of structure in language

Bridging the gap between symbolic and connectionist models of language processing

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Agnietenkapel
op woensdag 14 december 2011, te 12.00 uur

door

Gideon Borensztajn

geboren te Amsterdam.

Promotor:
Prof.dr. L.W.M. Bod

Co-promotor:
Dr. W.H. Zuidema

Overige leden:
Prof.dr. P.W. Adriaans
dr. F.P. Battaglia
Prof.dr. W. Bechtel
Prof.dr. L. Boves
dr. S. Frank
Prof.dr.ir. R.J.H. Scha
dr. P. Sturt

Faculteit der Geesteswetenschappen
Universiteit van Amsterdam

The research reported in this thesis was supported through a Vici-grant "Integrating Cognition" (nr. 277.70.006) to Rens Bod by the Netherlands Organization for Scientific Research (NWO).

Cover design by the author.
Printed and bound by Ipskamp drukkers.

ISBN: 90–5776–233–1

to my dear father

with love and admiration

Contents

Acknowledgments **xiii**

1 Preface **1**
1.1 The computational mind 1
1.2 Motivating questions and goals 2
1.3 Outline of the thesis 6

2 From Memory Prediction Framework to a neural theory of syntax **11**
2.1 Foundations of the Memory Prediction Framework 11
2.1.1 The hierarchical and topological organization of the cortex 12
2.1.2 Spatio-temporal encoding in the cortex 14
2.1.3 The hierarchical integration of temporal and perceptual information 15
2.2 Language processing in the hierarchical brain 16
2.2.1 Grammar induction in the MPF through merging and chunking 16
2.2.2 Object recognition is an interaction between bottom-up and top-down processing 18
2.2.3 Shortcomings of the MPF as a model of language processing 20
2.3 Neuro-biological solutions for the binding problem 21
2.4 Recursion, substitution and dynamic binding 23
2.5 Connectionist implementation of serial, dynamic binding 24
2.6 Localist networks and topology 26
2.6.1 The role of topology in a neural theory of syntax 26
2.6.2 Countercurrent systems, top-down and bottom-up networks 28
2.7 The role of episodic memory in language processing 29
2.7.1 The neurobiology of memory consolidation 32
2.8 A neural theory of syntax 35

2.9 A note on semantics 36

3 Symbolic approaches to language processing 39
3.1 Introduction to formal syntax and parsing techniques 40
3.1.1 Rewriting grammars 40
3.1.2 Automata and the Chomsky hierarchy 42
3.1.3 Parsing 44
3.1.4 Probabilistic context free grammars 45
3.1.5 Treebank estimation 46
3.1.6 Parser evaluation 46
3.1.7 Probabilistic left corner parsing 47
3.1.8 Variations to the PCFG statistical model 48
3.1.9 Data Oriented Parsing 49
3.1.10 Children's grammars grow more abstract with age 51
3.1.11 Chart parsing 52
3.2 Unsupervised grammar induction with CFGs 55
3.2.1 The Expectation Maximization Algorithm and the Inside Outside Algorithm 55
3.2.2 Bayesian Model Merging, Minimum Description Length . . 56
3.3 Limitations of the symbolic approach 58
3.3.1 Syntactic categories are prototypical and graded 59
3.3.2 Limitations of the probabilistic approach to syntax 61

4 Connectionist approaches to language processing 63
4.1 Introduction 64
4.2 The systematicity debate — critique of connectionism 65
4.2.1 Why connectionism? 67
4.3 Explicit representation of compositional structure 68
4.3.1 Recursive Auto-Associative Memory 69
4.3.2 Filler-role binding using the tensor product 70
4.3.3 The neural blackboard architecture 71
4.4 Recurrent, distributed networks 72
4.4.1 The Simple Recurrent Network (SRN) 73
4.4.2 Second connectionist reply to Fodor and Pylyshyn: distributed connectionism 74
4.5 Defining criteria for the systematicity of language 74
4.5.1 Weak and strong systematicity according to Hadley [1994] 74
4.5.2 A proposal for a concise definition of the systematicity of language 76
4.5.3 The importance of inductive bias for generalization 77
4.6 Systematicity and the Chomsky hierarchy 78
4.6.1 Evaluating the systematicity of formal grammars 79

4.6.2 A systematic model of language must have a structure bias that is at least as expressive as CFG . . . 82
4.7 Systematicity and the Simple Recurrent Network . . . 83
4.7.1 The structure bias of the SRN compares to that of an FSA 83
4.7.2 The SRN does not satisfy the context invariance criterion . 85
4.7.3 Locality of learning algorithms prevents true generalization of distributed patterns . . . 86
4.7.4 The systematicity of the SRN in the literature . . . 87
4.8 What the systematicity of language tells us . . . 87
4.8.1 The context invariance criterion . . . 88
4.8.2 Dynamic binding in syntax . . . 90
4.8.3 The recursive systematicity criterion . . . 93

5 The hierarchical prediction network 95
5.1 HPN architecture . . . 96
5.2 Representational power of HPN . . . 99
5.3 Parsing with the HPN grammar . . . 101
5.3.1 Probabilistic left corner parsing with HPN . . . 103
5.3.2 HPN node states . . . 103
5.4 Learning . . . 104
5.4.1 An example of the formation of abstract categories through self-organization of the topology . . . 106
5.5 Experimental evaluation . . . 106
5.5.1 Topology formation in an artificial language . . . 107
5.5.2 Recursive systematicity . . . 108
5.5.3 Topology formation in realistic corpora of children's speech 108
5.6 Chapter conclusions . . . 109
5.6.1 HPN and systematicity . . . 109
5.6.2 HPN versus RAAM . . . 110
5.6.3 Reaching the limits of HPN . . . 111
5.7 Neural interpretation of core components of the HPN model . . . 113
5.7.1 Connectionist implementation of a switchboard . . . 113
5.7.2 The tagging system . . . 115

6 Episodic grammar 121
6.1 Episodic memory . . . 121
6.1.1 Proposal for the representation of episodes as distributed traces in semantic units . . . 123
6.2 Episodic grammar — model outline . . . 124
6.2.1 The left corner episodic grammar . . . 126
6.2.2 Training the episodic grammar . . . 127
6.2.3 Statistical parsing with the episodic grammar . . . 128
6.2.4 Smoothing and binarization . . . 129

6.2.5 Evaluation and reranking . . . 131
6.3 Experiments and results . . . 132
6.3.1 Discontiguous episodes . . . 134
6.3.2 Shortest derivation reranker . . . 134
6.4 Relation to other work . . . 136
6.5 Chapter conclusion . . . 137

7 Parsing with episodic memory 139
7.1 An Earley-style probabilistic left-corner chart parser . . . 139
7.1.1 States of the left corner parser . . . 140
7.1.2 Probabilistic left corner shifting grammar . . . 141
7.1.3 Probabilistic left corner chart parsing . . . 142
7.1.4 Prefix probabilities . . . 145
7.1.5 An example that explains why inner probabilities are necessary . . . 146
7.1.6 Implementation issues . . . 147
7.2 Evaluation of the basic probabilistic LCSG chart parser . . . 148
7.3 The episodic left corner chart parser . . . 149
7.3.1 Most probable episodic parse . . . 150
7.3.2 Shortest derivation parse . . . 152
7.3.3 Shortest derivation left corner chart parser . . . 154
7.3.4 Implementation issues of the shortest derivation parser . . . 156
7.4 Experiments with the shortest derivation left corner parser . . . 158
7.4.1 Coarse-to-fine parsing and pruning . . . 159
7.4.2 Chapter conclusion and discussion . . . 159

8 Learning grammar through episodic memory consolidation 163
8.1 Introduction . . . 164
8.2 Treating language acquisition as a memory consolidation problem 164
8.2.1 Example 1: Lieven et al. [2003] . . . 165
8.2.2 Example 2: Marcus et al. [1999] and Marcus [2001] . . . 165
8.2.3 The use of analogy in computational models of grammar induction . . . 166
8.2.4 Towards a connectionist model of memory consolidation in language . . . 167
8.2.5 Discovering analogies via the principle of the shortest derivation . . . 168
8.3 The episodic-HPN model . . . 169
8.4 Predictions of episodic-HPN for memory consolidation . . . 172
8.4.1 What episodic-HPN says about the transformation (de-contextualization) from episodic to semantic memory . . . 172
8.4.2 The role of the hippocampus according to episodic-HPN . . . 172
8.5 Discussion . . . 175

8.5.1 Relation to other neural network models of memory consolidation . . . 175

9 General discussion and conclusions 179
9.1 Summary . . . 179
9.2 A possible role for HPN in cognitive linguistic research . . . 181
9.2.1 Modeling language acquisition with HPN . . . 181
9.2.2 HPN and construction grammar . . . 182
9.3 A reply to Jackendoff's challenges . . . 183
9.4 Relation to other work on unsupervised grammar induction . . . 185
9.4.1 HPN versus Inside-Outside . . . 185
9.4.2 HPN versus Bayesian Model Merging (BMM) . . . 187
9.4.3 HPN versus U-DOP . . . 188
9.5 Future work . . . 188

A Two case studies on the systematicity of the SRN 195
A.1 Case study 1: Elman [1990] . . . 195
A.2 Case study 2: Elman [1991] . . . 198
A.2.1 Fundamental reason why an SRN cannot generalize context free languages: analogy . . . 200

B HPN implementation issues 203
B.1 A deterministic and serial left corner parser for HPN . . . 203
B.2 Conversion procedure from (P)CFG to (P)HPN . . . 205

Index 225

Samenvatting 231

Abstract 235

Acknowledgments

First of all I want to express my utmost gratitude for having been so lucky to work with the two most wonderful supervisors one could possibly wish for, Jelle Zuidema and Rens Bod. While they each brought in specific qualities, both have been in their own way indispensible for my passing this tough ordeal called PhD.

As with so many PhD careers, this one has not been spared from its fair share of crises either. To Jelle and Rens, I want to thank you from the bottom of my heart for your patience and support, especially during hard times. It must not have been the easiest task for you to deal with all my peculiarities and stubborn, nonconformist convictions, and to keep my chaotic way of doing scientific research on the right track, but you have done a remarkable job, always being careful to find a delicate balance between pushing too hard and easing on me too much.

Jelle has been throughout the years my beacon in roaring scientific waters. His extensive and deep knowledge of subjects ranging from evolutionary biology to cognitive science to Artificial Intelligence and (computational) linguistics never ceased to astonish me, especially since he is a couple of years my junior. But more importantly I felt that I could always rely on his judgment, as it is profound and honest, guided by common sense rather than by ideological preconceptions. Jelle's unassailable instinct for what is good and what is bad science is invaluable in our field, and I learned from him to apply the same healthy criticism that I naturally have towards other research also to my own scientific practice.

Jelle has such an incredibly sharp mind that I have cut myself several times on it. In our frequent discussions he would always get right to the core of a problem that I was wrestling with, and suggest the most practical way to tackle it. This ability of his is all the more striking if you know that I often have a hard time to formulate my ideas in intelligible sentences, or even in a 'bag of words'. In those cases Jelle would show a rare capacity to read my thoughts, greatly facilitating our communication.

Rens has been a great inspirator, and he is the modern incarnation of the homo universalis, with broad interests and amazing encyclopedic knowledge ranging

from astronomy, the arts, history of the humanities and history in general, to linguistics and computer science. Rens took me totally by surprise when he hired me as a PhD on his Vici-project "Integrating Cognition" after I made a fool of myself during the job interview. Yet from my incoherent fumbling he managed to figure out that I had some vague ideas about doing something with the Memory Prediction Framework and language, and he gave it a chance.

Rens's intellectual openmindedness and (nearly insatiable) sciencific curiosity, free of conventional prejudices and barriers, provided me with an opportunity to follow my intuitions about language and the brain into largely unexplored territory. His enthusiasm and inexhaustable energy are proverbial at the ILLC, where we sometimes call him the whirlwind, because he gets more things done in an hour than most of us in a month. Thanks to Rens's frequent pep talks I kept believing in myself and in the project despite many setbacks. And thanks to his outstanding management skills I never had to worry about bureaucratic matters, and I could finish my PhD only one year overdue. Jelle and Rens, I hope I haven't burdened you too much with all my public and private preoccupations over the years, but now the time has come for me to release you, with a heavy heart, to a fresh draft of students.

Well, I am not going to tire you, the hypothetical reader of this thesis, much longer with a long list of acknowledgements, because I want you to be still in good shape before starting the first chapter. So I apologize beforehand to anyone who I might have overlooked, and I will just mention the following important persons:

Thank you, Federico Sangati and Markos Mylonakis (and before, Fernando Velázquez Quesada), for sharing a room with me at the university, for bearing with me despite my perpetual whining, for having funny and nonsensical as well as serious and helpful discussions, for many great games of ping pong which unfortunately came to an end when we moved to a new location, and for being friends.

Thank you, Stefan Frank, for being a great colleague and sparring partner in the systematicity debate. Thanks to our countless discussions Chapter 4 has greatly improved. Also, our short-lived mini-reading-group was much more fun than the CLS!

Thank you, Francesco Battaglia, for being one of the first people outside the ILLC to show interest in my crazy ideas. Thanks to Francesco and Alessandro Treves we established a collaboration with two PhD students from Trieste, Ritwik Kulkarni and Sahar Pirmoradian, who it was a pleasure to have as our guests, and with whom I have exchanged many ideas.

Thank you, Remko Scha, for inspiring philosophical discussions and for your kind willingness to read and comment on some very preliminary texts.

Many thanks to the staff members at the ILLC, Karin Gigengack, Tanja Kassenaar, Ingrid van Loon and Peter van Ormondt for doing such a good job taking all administrative issues out of our hands. Special thanks to Ingrid, Peter, Leen Torenvliet (who was director of the ILLC at the time) for their support, and

to Rosette van Raalte for her kind advices during a difficult period of my PhD. Thanks also to Tikitu de Jager for proofreading the first chapters of this thesis.

I would like to thank all the committee members for sacrificing much of their valuable time to reading my thesis, and for helpful comments. From the committee members I want to specially thank William Bechtel, whose beautifully written book "*Connectionism and the mind*" offers an excellent introduction to the issues and debates surrounding connectionist modeling of cognition and language, which allowed me to make a jump start with my research. Lou Boves also deserves special mention, because it was him, through some strange quirk of fate, who brought my attention to the work of Jeff Hawkins, which became central in my PhD research. I consider it a great honor that you all have agreed to be members on my PhD committee.

After the UvA moved to a modern building at Science Park, far outside the center of town, I preferred the intellectually stimulating environment of the Amsterdam coffee bars over that of the Science Park to carry out most of my research. Thank you cafe Krul, de Badcuyp, de Stadskantine, Pain Quotidien and above all 't Proeflokaal with its beautiful garden terrace for tolerating the silent presence of me and my laptop, mostly on a budget of a single cup of coffee or mint tea (and 220 Volts). You have unwittingly sponsored fundamental research.

Dearest of all, to my parents and closest family and friends, thank you for doing everything for me without ever asking anything in return, and for your love. Sorry that I was away on such a long journey. I hope all you got and will get from me is more than this lousy T-sis.

Amsterdam — Gideon Borensztajn
November, 2011.

Chapter 1

Preface

1.1 The computational mind

In 2004 a book on neural computation in the cortex was published, intriguingly titled "On Intelligence" [Hawkins and Blakeslee, 2004], that attracted a lot of attention within the cognitive science community. The book sketches an engineering perspective on the neocortex, inspired by a thorough analysis of the neural information pathways, that basically sees the brain as a massively connected pattern recognizer. The main claim of the book is that the function of intelligence and of memory is to *predict*, hence Hawkins dubbed his theory the "memory prediction framework" (MPF).

The pillars of the memory prediction framework are that (i) the neocortex extracts and encodes *temporal sequences* (of patterns) as categories in neural assemblies that correspond to cortical columns, (ii) the neocortex stores categories in a *hierarchical* fashion, (iii) as one goes up in the hierarchy, categories are formed that are progressively more *invariant* and more *temporally compressed*, and (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. Hawkins' functional analysis of the cortex is supported by a large body of neuro-biological research, particularly in the field of vision. (Some of this will be discussed in chapter 2, when I will dig a bit more deeply into the details of the MPF.)

It struck me that the ideas outlined in the MPF offer good prospects for a theory of language processing: the idea that categories higher in the cortical hierarchy are temporally compressed corresponds well to linguistic intuitions about phrase structure. Indeed, what is a high level syntactic category if not a compressed representation of a sequence of words? Further, the interleaved integration of the temporal and the sensory dimensions (at all levels of the cortical hierarchy), which is one of the principle features of the MPF, is analogous to syntagmatic and paradigmatic processes (processes concerning positioning and substitution respectively) in syntactic theory (more about this in section 2.2.1). However,

the book and ensuing technical publications [George and Hawkins, 2005, 2009] are tailored towards visual processing, and no linguistic applications have so far appeared. The reason is, in my opinion, that the MPF in its current form is not yet suited for productive language processing, due to certain shortcomings of the framework. In section 2.2.3 I will suggest several improvements and extensions of the MPF, and I will propose that the modified version of the MPF may serve as the basis for a neural theory of language processing.

Curiously, the insights from the MPF are very much at odds with current trends in connectionist theories of language, although both frameworks aspire to be neurally motivated. In recurrent distributed networks, probably the most popular connectionist platform for modeling language processing, syntactic categories are not recognized, nor is hierarchical structure, and a quite different conceptualization of temporal processing is embraced than by hierarchical compression. A substantial portion of this thesis will be dedicated to a critical review of the literature on connectionist language processing and systematicity. I will argue, against received wisdom, that 1) the criteria of systematicity of language that are currently popular among connectionists are inadequate, and 2) conventional distributed connectionist models do not satisfy a more concise set of systematicity criteria, that incorporates phrase structure, and hence are ill suited for modeling natural language acquisition. On the other hand, I will identify within the Memory Prediction Framework some critical neural mechanisms that could be incorporated by the brain to solve the problem of constituent structure representation. These ideas lead to a proposal for a novel type of connectionist network, that is able to learn a grammar and parse sentences, and that is based on an architecture that complies with the hierarchical organization and information flow in the cortex.

1.2 Motivating questions and goals

Why would anyone care to inquire into the neural basis behind linguistic computation? Given that so little is known about the neural instantiation of language, how can one expect neuro-biological considerations to help in understanding how people produce comprehensible sentences? There are different answers to these questions, depending on who you ask.

For the linguist with a background in formal grammars it should be interesting to understand how typical linguistic constructs and operations can be implemented in the brain.

In *phrase structure grammars*, the basis of many linguistic formalisms, the grammar consists of abstract syntactic categories (e.g., NP, VP), and a set of 'rewrite rules', which combine categories into larger structures, and eventually into a sentence. For example, the sentence *The woman read the book* might be represented by the tree in figure 1.1, using the following rewrite rules in a

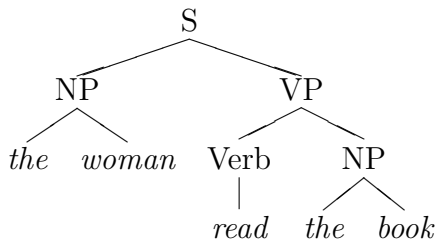


Figure 1.1: Example of a parse tree

derivation, starting from the top of the tree: $S \rightarrow NP\ VP$, $NP \rightarrow the\ woman$, $VP \rightarrow Verb\ NP$, $Verb \rightarrow read$ and $NP \rightarrow the\ book$. Typically, such models are formulated at an abstract 'competence level', and are indifferent about the processing level. Yet, certain questions about the neural foundations of language deserve the attention of the linguist, but cannot be phrased using the formal apparatus of competence grammars. For instance, how are syntactic categories and rewrite rules (or their equivalents) instantiated in the brain? How is a rewrite rule selected and accessed in the brain? How are the symbolic syntactic variables acquired, and where does their global scope derive from? How can a parse be neurally processed, through local interactions alone? How can the brain represent parse trees of unbounded depth using only limited brain space?

To advance insight into these questions one must move to the processing level. The added value of integrating a computational model of language with a neural architecture (e.g., by inspiration from the MPF) is that it puts physical constraints on the implementation of the grammar. For instance, complex syntactic operations, such as center embedding (i.e., nesting a phrase in the middle of another phrase), are necessarily executed by local processes that employ the wetware of the brain, and parse trees of arbitrary depth must be represented in a restricted space. Some solutions that are uncontroversial in formal linguistics, such as the use of variables, become real challenges, and unexpected questions may pop up if the innateness assumptions from generative grammar are dropped (e.g., how does substitutability emerge?).

Jackendoff [2002] presents what he sees as the four major challenges for a neural theory of language:

1. *The problem of representing variables* concerns the idea that it seems necessary that (systematic) knowledge of language be encoded as abstract relations between typed variables. This is particularly important to explain the combinatorial productivity and the systematicity of language, that is a person's ability to produce and understand an unbounded number of novel sentences based on a limited number of observed sentences. Fodor and Pylyshyn [1988] argue that for systematic language processing the cognitive system must be able to perform operations over symbols, or variables.

Yet, traditional neural networks cannot encode variables for principled reasons [see e.g., Marcus, 2001] (see also section 4.7.3). The debate about the systematicity of language will be covered extensively in section 4.2.

2. *The massiveness of the binding problem* refers to the fact that in order to understand a sentence, phonological, syntactic and semantic levels must be combined, and within each structural level substructures, such as phrases, must be bound. Given that people can understand and produce a near infinite number of sentences, if every sentence required a dedicated neuron that binds the substructures, the amount of binding needed would be massive.

3. *The asymmetry between binding in working memory and binding in long term memory (LTM)* refers to the question how the brain deals with the storage of productive versus fixed linguistic expressions. There exists a discrepancy between expressions, such as 'lift the shovel', that are produced flexibly on-line in working memory and fixed idioms, such as 'kick the bucket', or reusable multi-word constructions that are apparently encoded as a whole in a person's long term memory, or lexicon.

4. *The problem of two* arises because, as Jackendoff [2002, p.58] argues, in order to be able to understand a sentence, all bindings, at all different structural levels must be functionally present at once. The question arises how the brain can deal with the multiplication of representations of the same word, that seems to be required whenever some word occurs more than once within a sentence.

In the next chapter (section 2.8) a neural theory of syntax will be developed, based on which an answer to Jackendoff's four challenges can be formulated (in section 9.3). Chapter 5 and subsequent chapters are dedicated to a computational implementation of this theory, which I call the 'hierarchical prediction network' (HPN).

For a second group of language researchers, of connectionist persuasion, the ideas presented by the MPF and further explored in this thesis offer an alternative perspective on neural processing, which challenges many of the standard assumptions of connectionism. Connectionism holds that a uniform, domain general cognitive architecture can explain (specifics of) language, as well as behavior in other modalities, and it aims at designing general purpose neural networks, which are simplified models of the brain, that can perform linguistic and other tasks. The fundamental reason why I adopted a connectionist approach to syntax acquisition in this thesis has to do with learnability: crudely stated, neural networks are capable of acquiring the meaning of their primitive units from the external environment, whereas symbolic models presuppose innate meanings of their primitives (e.g., syntactic variables). More about this will be said in section 4.2.1.

Connectionists have quite a different agenda than the first group of linguists, since in general (exceptions like Prince and Smolensky [1997] notwithstanding) they do not accept many of the foundational assumptions of formal linguistics anyway. One of the main claims of this thesis is that existing connectionist models have fundamental problems explaining systematic linguistic behavior (see section 4.5.1), and new ideas are needed to make connectionist models more congruent with insights from traditional linguistics (specifically the Chomsky hierarchy, section 3.1.2). It will be shown that this goal can be achieved through a tighter integration of connectionist models with the cortical architecture that is suggested by the MPF.

A third group of linguists that might find interest in this thesis are those that identify themselves with *construction grammar* [see e.g., Goldberg, 2003, 2006, Fillmore and Kay, 1987]. In construction grammar (CG) the primitive units of language production are *constructions*, which are stored associations between a semantic frame and a syntactic pattern, which may consist of one or more words with optional variable slots. In Radical Construction Grammar [Croft, 2001, 1991] it is claimed that all syntactic patterns in language are in fact constructions, and that syntactic categories are defined relative to constructions with local validity or scope. Learning in CG consists of the gradual acquisition of a structured inventory of constructions, a *constructicon*, where the constructions are of various sizes and varying degrees of complexity and abstractness [see e.g., Goldberg, 2006, Tomasello, 2003].

Empirical studies in this tradition [e.g., Peters, 1983, Tomasello, 2001, van Kampen, 2003] emphasize a dynamically changing grammar that follows a trajectory, during early development, from simple and concrete to complex and abstract constructions. While early child language is item-based in nature (often organized around so-called verb-islands), in subsequent stages children start breaking down the item-based constructions, introducing variables in slots, such as in Where's the X?, I wanna X, etc. According to Tomasello's Usage Based Grammar (UBG) [Tomasello, 2003] the acquisition of constructions with variable slots forms the beginning of abstraction and category formation, and it marks the beginning of grammar [Tomasello, 2000a].

These findings suggest that underlying children's language production is a syntactic representation in which the primitive building blocks can be of variable size, form and level of abstraction. A grammar formalism that can accommodate such flexibility is the family of of Tree Substitution Grammars (TSG) (section 3.1.9), and particularly Data Oriented Parsing (DOP; [e.g., Bod, 1998, Scha et al., 1999]). In section 3.1.10 I will briefly present published work that is based on the DOP framework, and that shows that Children's grammars grow more abstract with age [Borensztajn et al., 2009a]. Yet, the major part of this thesis is concerned with the more fundamental question of what constitutes the neural basis behind syntactic constructions.

This thesis implements a computational model of a constructicon, and models

its acquisition as a growing, self-organizing network that gradually evolves from a loose collection of concrete sentence fragments to an integrated network of abstract adult-like grammar rules. The model illustrates, in a neurally plausible manner, how constructions with variable slots — open positions where variable input can be inserted — become progressively more abstract (section 5.4.1), providing a computational basis for the usage based acquisition process. Finally, by implementing the constructicon as a neural network it can identify neural processes that underly language acquisition and processing: in section 2.7 and Chapter 6 I will discuss how the constructicon, consisting of a mix of concrete, multi-word expressions and abstract rules, can be understood in terms of the human memory system, that stores a combination of episodic and semantic memories in the language domain.

1.3 Outline of the thesis

This thesis is divided into several parts, which is a consequence of the fact that the research presented resides somewhere between connectionism and statistical parsing, and between rule-based and exemplar-based approaches to language processing. One of the major objectives of this work is to show, both theoretically and empirically through modeling work, that the perceived absolute dichotomy between connectionist and symbolic approaches to language processing, as they are currently formulated, is based on false assumptions on both sides, and that by clearing existing misconceptions (e.g., concerning systematicity, variable operations, recursion and binding in the brain) the gap can be bridged. In this respect, Chapter 2 presents a neural perspective on language processing in the cortex, based on the Memory Prediction Framework, that emphasizes its hierarchical and localist nature, and that, if taken seriously, leads one to question the standard assumptions (i.e., distributed and recurrent processing) of mainstream connectionism.

Starting from Chapter 5 I will introduce, in two stages, a connectionist model of syntactic processing, called the "hierarchical prediction network" (HPN), that integrates elements from neural networks and statistical parsing, and thereby forges a synthesis between these frameworks. Subsequent chapters investigate the neural basis of exemplar based models of language, and the thesis reaches its conclusion in Chapter 8 with a proposal for an exemplar-based connectionist model of sentence processing, episodic-HPN, which is an extension of HPN with episodic memory.

The leading motives throughout the thesis are questions concerning the fundamental problems of language acquisition. I will try to answer these by looking at the parallel between language acquisition and memory formation: the first part of the thesis, and the first version of the HPN model, focuses on learning syntactic categories and rules and the construction of a semantic memory in which abstract

categorical relations are laid down in the network topology of HPN; the second part, and the final version of HPN, focuses on sentence processing with exemplars, and the transition from concrete exemplars to abstract rules, as a process of episodic to semantic memory consolidation.

- **Chapter 2** introduces the Memory Prediction Framework (MPF), and it contributes some original ideas intended to accommodate a neural theory of language processing within the existing framework. The chapter tries to link insights from the neuroscience literature to natural language processing by exploiting analogies between language and visual processing. For instance, in section 2.3 I discuss current research on dynamic binding in visual processing, and I propose that a similar mechanism is responsible for substitution in syntactic parsing. In section 2.6 I discuss localist connectionist networks, such as the Kohonen network, and the possible role of topology in representing graded syntactic category membership. In section 2.7 I discuss the possible role of the episodic memory system in exemplar-based language processing and acquisition, and finally in section 2.8 the ideas of this chapter are summarized in a neural theory of syntax, inspired by the MPF.

- **Chapter 3** presents an introduction to formal syntax and parsing techniques with an emphasis on left corner parsing. The chapter gives some basic background on probabilistic context free grammars and statistical natural language processing; some variations of the PCFG statistical model are mentioned, as well as an exemplar-based computational model of syntactic processing, Data Oriented Parsing (DOP) [Bod, 2003]. An application of DOP for finding multi-word constructions in children's speech is discussed [Borensztajn et al., 2009a]. The Earley chart parser is discussed in some detail, as well as its probabilistic version, because they will play a role in the HPN model. Then I spend some sections on unsupervised learning (induction) of grammar, focusing on two approaches: Bayesian Model Merging (BMM) and the Inside Outside algorithm. In the final section I discuss shortcomings of the formal (and statistical) approach to syntax, particularly with respect to explaining first language acquisition, motivating the decision to embark on connectionist language processing. Section 3.1.10 contains a summary of the following publication:

  [**Borensztajn et al., 2009a**] Children's grammars grow more abstract with age. Evidence from an automatic procedure for identifying the productive units of language. *Topics*, 1 (1): 175-188, 2009.

- **Chapter 4** presents a critical review of (a part of) the literature on connectionist language modeling, in the context of the systematicity debate. The chapter starts with a discussion of a class of connectionist models with

a built-in capacity for compositional structure representation [e.g., Pollack, 1988, Prince and Smolensky, 1997]. I argue that the way these models deal with structure representation and binding is in principle and intrinsically not connectionist. After a short survey of the connectionist literature on systematicity, during which I discuss Hadley's [1994] criteria for weak and strong systematicity, I will propose in section 4.5.1 that a more precise, and stricter, set of criteria is necessary to evaluate systematicity in connectionist models: the context invariance and recursive systematicity criterion. In section 4.8.1 I argue that the Simple Recurrent Network (SRN) [Elman, 1989], which is commonly regarded as the classical example of a connectionist model capable of systematic language processing, fails both criteria. To back up my claims, in appendices A.1 and A.2 I try to expose the fallacies in the literature about the systematicity of the SRN. Finally, in section 4.8, I discuss the implications of the reformulated criteria for the systematicity of language, for contemporary connectionist language research. I propose a conceptual revision of the standard assumptions about connectivity in the brain as well as fundamental changes in the design of neural networks for language.

- **Chapter 5** introduces the basic HPN model. The core concepts and components of HPN are discussed and motivated from the MPF and the neural theory of language. I provide formal definitions of HPN, and discuss some formal properties, such as the fact that HPN subsumes PCFG's. Subsequently, I discuss a cognitively plausible parsing algorithm, implemented in HPN and based on the Left Corner Parser, and the incremental learning algorithm of HPN that is complementary to parsing. Section 5.5 presents preliminary experiments with the first version of HPN, and section 5.6.3 discusses its limitations (e.g., a limited possibility for conditioning parser decisions on context). The conclusion motivates the work on episodic grammars described in subsequent chapters leading towards a version of HPN with integrated episodic memory. The content of this chapter is partially extracted from the following publication:

  [**Borensztajn et al., 2009b**] The hierarchical prediction network: towards a neural theory of grammar acquisition. *Proceedings of the* 31^{st} *Annual Conference of the Cognitive Science Society.*

- **Chapter 6** focuses on the question of how exemplar-based linguistic knowledge, needed for contextual conditioning in parsing and for learning, is represented in the brain. I propose that an episodic memory for previously processed sentence analyses is responsible for the brain's ability to appeal to sentence context, and I present a novel idea about the nature of the relation between semantic and episodic memory in the brain. This idea is

implemented in a probabilistic episodic grammar, and evaluated as a syntactic parser on corpora of natural language. I explain how, in the symbolic and supervised case, the episodic grammar is trained from a treebank, and how a probability model can be defined based on priming and retrieval of episodic sentence memories. The content of this chapter is partially extracted from the following publication:

[**Borensztajn and Zuidema, 2011**] Episodic grammar: a computational model of the interaction between episodic and semantic memory in language processing. *Proceedings of the* 33^{rd} *Annual Conference of the Cognitive Science Society.*

- **Chapter 7** introduces the probabilistic left corner shifting grammar (PLCSG), and a PLCSG chart parser that will be used with HPN. It implements a dynamic algorithm for the efficient computation of sentence probabilities and prefix probabilities. I show how the PLCSG chart parser can be extended for episodic parsing, and I develop a Viterbi algorithm for finding the shortest left corner derivation. Subsequently, the episodic left corner parser is evaluated for precision and recall on a realistic natural language corpus, and a qualitative analysis of the resulting shortest derivation parses is presented.

- **Chapter 8** presents the episodic-HPN model, an extension of HPN with an episodic memory. Here the parallel between language acquisition and memory consolidation will be crystallized. The chapter starts with a short overview of the literature on artificial language learning with infants. It is argued that learning the structure of a language (a grammar) is greatly facilitated by the extraction of structural analogies between stored sentence pairs, and that for this to happen an episodic memory of processed sentences is required. The episodic-HPN model shows how the transition between episodic (sentence) memory and semantic memory (an abstract grammar) can be guided by the principle of the shortest derivation. The model implements an exemplar-based connectionist parser and a full probability model, that is able to condition on the entire sentence history, hence is suitable for processing sentences of natural language.

- **Chapter 9** contains a general discussion and conclusions, a retrospective overview of the thesis, a reply to Jackendoff's challenges and many suggestions for future work.

Chapter 2

From the Memory Prediction Framework to a neural theory of syntax

In this chapter I will highlight several details from the Memory Prediction Framework (MPF) [Hawkins and Blakeslee, 2004], which inspired the current work. Whereas Hawkins and Blakeslee [2004] focus mostly on visual processing by the cortex, I will point out some striking analogies with theories of language processing, supporting the idea that there exists a uniform cortical mechanism that underlies categorization and processing within both the visual and the syntactic domain. Subsequently, I will identify and briefly motivate certain neural components that are lacking from the MPF, but which would be needed to successfully deal with productive language processing. Together with the MPF these components make up a neural theory of syntax, outlined in section 2.8, and they will be the ingredients for the computational model of syntax that will be developed in the remainder of this thesis.

2.1 Foundations of the Memory Prediction Framework

The MPF is founded on a philosophy about the origin of human knowledge that is nowadays commonplace in cognitive science [e.g., see Crick and Koch, 1998]. According to this philosophy, the evolutionary function of the cortex is to record

causal relations, which can help the organism in making predictions based on observations from experience. To do so it must partition the continuous stream of information that enters the senses into discrete objects, and construct an internal world model to interpret relations between objects.

> The hierarchical structure of the cortex stores a model of the hierarchical structure of the real world. The design of the cortex and the method by which it learns naturally discover the existing hierarchical relationships in the world [Hawkins and Blakeslee, 2004, p.125].

Memory is intimately related to planning: given a memory system that can store causal or temporal relations between events, chances for survival are enhanced, because it enables the organism to anticipate or predict likely events in the future, and the likely outcomes of its own actions.

> [E]volution discovered that if it tacks on a memory system (the neocortex) to the sensory path of the primitive brain, the animal gains an ability to predict the future. [Hawkins and Blakeslee, 2004, p.99]

In [Hawkins and Blakeslee, 2004, Chapter 4], Hawkins explains how the neocortex does this. The following four important observations about the neocortex convey its basic function as a *temporal pattern recognizer*, and distinguish it from a conventional computer. To recapitulate,

1. The neocortex stores everything as temporal sequences of patterns.
2. The neocortex stores patterns in invariant form.
3. The neocortex stores patterns in a hierarchy. Categories are progressively more invariant and temporally compressed as one goes up to higher levels in the cortical hierarchy.
4. The neocortex recalls patterns auto-associatively.

2.1.1 The hierarchical and topological organization of the cortex

The architecture of the MPF is modeled after the hierarchical organization of the visual cortex, and since this plays an important role in this thesis I will provide the essential neuro-biological background here. As one moves vertically through the ventral pathway of the visual cortex (which leads from V1, via V2 and V4 to the inferotemporal cortex (IT)), a hierarchy of increasingly complex and location invariant representations of visual information is formed, while at the same time the receptive fields of the columns increase in size [see e.g., Felleman and van Essen, 1991, Kobatake and Tanaka, 1994]. At the lowest level neurons selectively

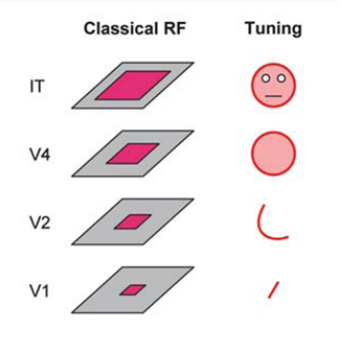


Figure 2.1: The visual hierarchy. From [Roelfsema, 2006]

respond to simple features, such as line segments at specific orientations, while adjacent neurons respond to orientations that are slightly tilted in such a way that orientation preferences gradually vary along the surface of the cortex. At higher cortical levels *complex* cells can be found that respond to more complex features, such as specific angles formed by combinations of line segments [Hubel and Wiesel, 1968]. Other neurons respond to stimuli of increasingly complex and irregular shapes [e.g., Maunsell and Newsome, 1987], as well as conjunctions of features from different modalities, for instance shape and color (see Figure 2.1).

Recordings in humans with electrode implantations (performed in epileptic patients) have consistently yielded discoveries of so-called *grandmother cells* that respond to specific concepts, and do so invariantly across different modes of presentation of the stimulus. For instance, Kreiman et al. [2002] reported a cell that responded to any image with Bill Clinton in it, whether it was a line drawing of a laughing Clinton, a painting, or a photograph of Clinton. Cells that are sensitive to famous actors (e.g., Jennifer Anniston), to the Eiffel tower and many other people and objects have also been found [Quian Quiroga et al., 2005].

The neurons within a sheet of cortex are organized in vertical *columns*, which are generally viewed as the basic computational units of the cortex [Mountcastle, 1997]. Within a vertical column in the cortex the neurons are typically tuned to the same signal. For instance, within a single 'orientation column' all neurons have the same orientation preference. Many other regions in the neocortex are organized in topographic maps, where similar feature values are encoded by adjacent neuronal columns within a brain region. Topologies have been demonstrated for visual inputs, tactile inputs (where the position on the skin surface is mapped to a somatotopic map), and in the auditory cortex, where tonotopic maps represent acoustic signals ordered according to their pitch frequency.

In sum, the sensory cortex is organized horizontally in topographic maps of feature detecting columns, and vertically in a hierarchy of regions with progressively fewer columns that respond to increasingly complex and abstract features with larger receptive fields. This results in a pyramidical structure, as schemati-

cally illustrated in Figure 2.2.

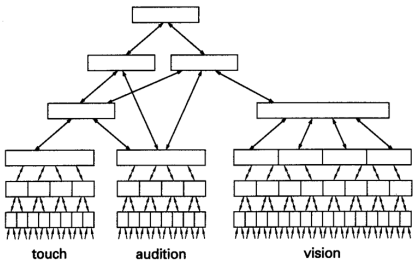


Figure 2.2: Schematic diagram of the cortical hierarchy according to MPF. Blocks represent cortical regions. (Reproduced from [Hawkins and Blakeslee, 2004].)

At the sub-columnar level one can distinguish six layers, L1, L2, ..., L6, in each sheet of cortex.Within each column (and at every level in the hierarchy) two opposing currents of information flow can be distinguished, one ascending (feed forward) and one descending (feed back) [Felleman and van Essen, 1991]. The upward flow enters at L4, with input converging from many regions below, and exits to a higher level via L3 and L2, while skipping L1. While the upward information mostly stays within a single column, the downward flow of feedback information is divergent, starting at L1, where lateral axonal connections spread across many columns, exciting cells in L2, L3 and L5, and exiting via L6 to the cortical level below.

To make sense of cortical computation at the algorithmic level, Hawkins and Blakeslee [2004] analyzed the information flow through the cortical hierarchy from an engineering point of view. What makes the MPF particularly interesting is that it tries to explain the functional connections between regions from different levels in the hierarchy, how invariant representations come about, and how *temporal* relations are encoded in the cortex.

2.1.2 Spatio-temporal encoding in the cortex

Since all visual, auditory and tactile experience involves change or motion, it makes perfect sense that the cortex memorizes temporal sequences of predictably correlated events. Even the memory of a face consists of an (unordered) sequence of fixations (e.g., on nose, eyes, and mouth) related via rapid eye saccades.

Suppose an observer perceives a tilted line moving towards him: then the orientation detected by the sensory apparatus in successive time steps gradually changes value, and each snapshot of the line activates a different orientation column in a cortical region. Sequences of successively activated orientation columns that reliably predict future observations are stored as prototypes within the region.[1] In a world where motion and shape are usually smooth and continuous,

[1]Details of how this might occur are described in Hawkins and Blakeslee 2004, p.146. In

columns that encode slightly different values of a feature are often activated in succession within a sequence, and will develop strong connections. Therefore, temporal or causal relations in the world play a role in structuring the cortical topologies, and they explain how neatly arranged feature topologies have emerged in the brain. I will argue in section 2.6.1 that this is also the case for the language domain: in syntax a topology is constructed that expresses predictable temporal relations between words and phrases.

The cortex not only encodes temporal sequences, it also binds simultaneously occurring features (e.g., position, shape, orientation) that converge to a column from different regions into a single unit, or *chunk*, in a higher cortical level. The next section explains in detail how the alternation between temporal compression into prototypical sequences and conjunctive feature binding into prototypical patterns leads to the incremental formation of highly invariant cells, such as the Bill Clinton cell, in the top level of the cortex.

2.1.3 The hierarchical integration of temporal and perceptual information

Hawkins and Blakeslee [2004] propose that hierarchical extraction of temporal and categorical invariant structure proceeds in two alternating steps, that are repeated in every level of the cortical hierarchy. In the first step, the most frequent prototypical conjunctive patterns composed of sequences from lower regions are extracted, and stored in the columns of the region above (see Figure 2.3).

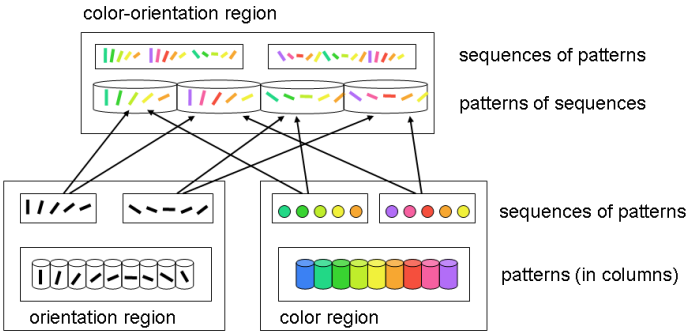


Figure 2.3: The process of hierarchical temporal feature extraction

In the second step, the most frequent prototypical temporal sequences of successively firing columns within a region are extracted, and encoded in the L1

short, through a thalamic feedback loop signals from previously active columns are relayed to the L1 neurons of a column, which have axonal connections to many other columns in the region. The synaptic connections from the L1 neurons to other columns are strengthened when they are simultaneously active, resulting in the encoding of temporal sequences.

neurons that connect to all active columns in a region. The stored sequences from multiple regions in turn serve as an alphabet from which, in the next iteration, prototypical patterns are constructed in columns of the next higher level, and so on. In sum, learning of invariant structure consists of an alternation between storing patterns of sequences and sequences of patterns.

A crucial question is how (temporal and spatial) invariance arises in higher regions of the cortex, leading for instance to the 'Clinton cell'. In order for the features to become progressively more invariant, a *constant* pattern must be relayed between the levels in the hierarchy during learned sequences. According to Hawkins, the answer is that in order to obtain an invariant representation of the sequence a mechanism is in place that abstracts over the details of the sequence, and only relays a 'name' for the sequence (i.e., a pointer) to the higher level region.[2] Within the higher level the name is regarded as a building block for a larger pattern, which does not care about the constitution of the name's referent. In fact the name (or pointer) is itself stored in the (L6 cells of the) higher level, where it is embedded within the context of a larger pattern of which it is a component. This mechanism is very important, because only from the embedding in a higher context can a category derive its extensional meaning. Meanings of categories cannot exist in isolation, but are *inferential*, which means that they are indirectly connected to the meanings of all other categories in the brain. I will refer to this operating principle as *encapsulation*. Encapsulation at successive levels in the cortical hierarchy ensures that the brain can make higher order abstractions, or generalizations over generalizations. In section 4.8.1 I will argue that an ability for encapsulation is a desirable property for connectionist models of language, which is lacking from distributed networks.

2.2 Language processing in the hierarchical brain

2.2.1 Grammar induction in the MPF through merging and chunking

What started this research project was the observation that there exists a remarkable parallel between the operations that were identified in the previous section as the basic operations employed by the cortex to hierarchically extract invariant sequences, and certain algorithms for the acquisition of syntactic categories that are used in the field of computational linguistics. One such algorithm for unsupervised grammar induction is Bayesian Model Merging (BMM) [Stolcke and Omohundro, 1994, Borensztajn, 2006b]. I will only briefly discuss BMM here; a

[2]Hawkins and Blakeslee [2004, p.150-2] suggest a complex mechanism for how the 'names' are encoded in L2 cells, which stay active as long as the actual stimulus matches the expectation of the sequence. Only in case of a mismatch are the L2 cells inhibited such that the details of the sequence rather than its name are passed to the higher level.

more formal exposition of BMM will be given in section 3.2.2 in the context of unsupervised grammar induction. In BMM two operators, *merge* and *chunk*, are used for the extraction of hierarchical structure from flat sentences:

- The *chunk* operator concatenates or 'chunks' repeating patterns (sequences), such that the pattern may be stored just once in its entirety, and can then be accessed via pointers (in a so-called *syntagmatic* process).
- The *merge* operator creates generalizations by forming disjunctive groups (categories) of patterns that occur in the same contexts (a *paradigmatic* process).

The BMM algorithm searches through the space of context free grammars, while at every step in the search it makes a small change to the grammar using one of these two operators. Merging and chunking alternate, until neither merging nor chunking improves the chosen objective function.

This procedure bears significant resemblance to the procedure for extracting invariant categories in the MPF, which was described previously. Transposed to the language domain, the same mechanism that Hawkins describes for visual category induction should be capable of syntactic category induction. If one assumes that words are assigned to the input units in the bottom layer of the MPF hierarchy (e.g., on a two-dimensional topological map), then extraction of prototypical sequences over patterns of words (or phrases) corresponds to chunking, and extraction of prototypical patterns over sequences corresponds to merging (of sequences with common contexts). Such considerations reveal that the hierarchical mode of operation of the cortex proposed by the MPF may provide an ideal machinery for grammar induction, as it does for discovering other kinds of hierarchical relations in the world.

Merging and chunking achieve generalizations in perpendicular dimensions. Merging results in generalization over the lexical dimension, or any other sensory dimension that is topographically encoded in a cortical region, and chunking is generalization over the temporal dimension. Merging and chunking are the basic operations of the brain (and, in my opinion, by no coincidence also correspond to the basic connectives in logic, $\vee$ (OR) and $\wedge$ (AND)). Together, merging and chunking interact in such a way that predictable temporal relations over observed sentences are distributed over the topology of the cortex, analogous to how the temporal distribution in the visual input is reflected in the topological distribution of visual features, such as orientation.

Interpretation of syntactic constituents

The above ideas imply the existence of categorical representations in the cortex that are neural correlates of syntactic constituents. In analogy to visual categories, I propose that 'syntactic' neural assemblies exist that represent temporally

compressed and invariant sequences of words. Higher order constituents can be learned by the cortex via repeated application of 'merging' and 'chunking' operations. Unfolding a 'syntactic' assembly in the temporal dimension corresponds to the expansion of a constituent to its right hand side within a rewrite rule (compare for instance the expansion $VP \rightarrow V\ NP$). Like syntactic constituents, invariant categories can unfold in multiple ways. Yet, unlike conventional symbolic categories such as NP or VP, the invariant neural representations of syntactic categories do not behave like global variables, but like prototypical categories, with local scope and graded category membership.

2.2.2 Object recognition is an interaction between bottom-up and top-down processing

According to the MPF visual object recognition is not a simple matter of bottom-up classification, but it involves an interaction between top-down *predictions* and bottom-up activation, which are matched at all levels in a so-called 'hypothesis-and-test' cycle. Predictions are executed by *unfolding* invariant representations of temporally compressed sequences.[3]

Since the purpose of maintaining abstract invariant representations is that they can be executed in many ways, there is a question how an invariant and unspecific representation selects to which specific prediction it is unfolded. To select a specific prediction, additional bottom-up information must be combined with the non-specific top-down information. For instance, when you want to catch an approaching ball, the first thing the brain does is to activate a highly invariant motor scheme for 'ball catching', but as the catching process unfolds progressively finer motor actions and perceptional stimuli interact in tight coordination to produce a specific catch that is fine-tuned to the situation. Thus, there is a continuous dynamical interaction between bottom-up sensory input and top-down invariant predictions, during which moment after moment increasingly specific predictions are compared with the sensory input, and adjusted in case of conflict, until eventually an equilibrium is reached between top-down expectations and bottom-up stimuli.

Computational techniques for discovering the hierarchical structure of events unfolding in time are much studied in linguistics, where it is known as *parsing*. Algorithms that perform a search through a space of possible structural analyses (of a sentence) are known as *parsers*. A similar 'hypothesis-and-test' search procedure as described above is implemented in *Left Corner Parsing* (LCP) [Rosenkrantz and Lewis II, 1970], a cognitively plausible parsing strategy that combines top-down and bottom-up processing, and proceeds incrementally through the sentence, from left to right. Left corner parsing will also be used as

[3]Similar ideas about the 'hypothesis and test' approach to object recognition had been expressed earlier [e.g., Ullman, 1991], but did not involve the temporal component.

the search algorithm of choice in the HPN model developed in this thesis; two versions of the algorithm will be described in detail in Chapter 7 and in Appendix B.1.

Nowadays parsing techniques are increasingly successful at solving complex visual object recognition tasks [Ullman, 2007, Han and Zhu, 2005], supporting the idea that there exists a uniform cortical algorithm that underlies both visual and linguistic processing.

The fragment-based hierarchical object recognition model

As an illustration of the parallel between visual and linguistic processing I will briefly discuss a computational model for visual object recognition proposed by Ullman [2007]. Ullman's model is in the spirit of the MPF, as it implements the principle of top-down versus bottom-up expectation matching, but it is somewhat less complex than the Bayesian algorithm proposed by George and Hawkins [2005].

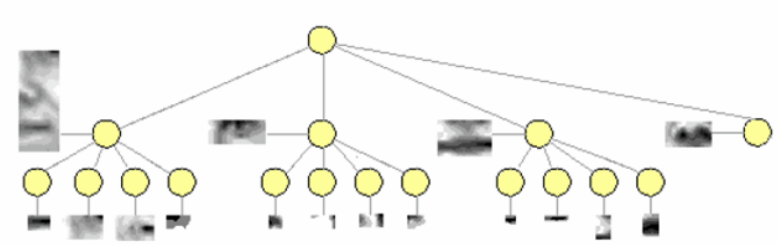


Figure 2.4: Fragment based hierarchical object recognition. Reproduced from [Ullman, 2007]

In the 'fragment-based hierarchical object recognition' (FHOR) model [Ullman, 2007, Ullman et al., 2002, Lifshitz, 2005] a visual object is decomposed into informative sub-components, or 'fragments', resulting in a hierarchical object representation (see Figure 2.4). After informative fragments of various sizes and complexity have been extracted from a corpus of images, the classification of a novel image amounts to an interaction between bottom-up grouping of image regions, based on perceptual similarity, and a top-down segmentation process, based on class membership prediction. Visual object recognition thus involves the construction of visual parse trees.

Note that the FHOR model makes use of visual categories, or constituents (the fragments), that figure in the 'rules' of a 'visual grammar'. For the classification of novel images invariant visual categories must be formed (e.g., to recognize a face in a novel image one needs an abstract category of a mouth, and eyes). As in language, abstraction is critical for productivity and for the interpretation of unseen objects.

One of the abstraction mechanisms used in FHOR is based on common context. "If two fragments are interchangeable within a common context, they are likely to be semantically equivalent" [Ullman, 2007, p.61]. Note that this is the same principle that was described in section 2.2 as a mechanism to induce syntactic categories of words from their sentential context (in grammar induction this is called 'merging'). Another mechanism used in the FHOR model is that visual fragments that often occur together are lumped into larger fragments and stored as a single unit. This, too, is a familiar principle in language learning, known as 'chunking'.

Yet one important aspect is ignored in most of the visual object classification literature, and that is the role of time. Precisely the fact that the temporal component of cortical processing is emphasized in the MPF makes it an interesting framework for language processing.

2.2.3 Shortcomings of the MPF as a model of language processing

Despite its merits, the MPF in its current form is not suited to explain some important aspects of how the cortex deals with language processing. One question concerns the representation of recursively embedded sentences inside an architecture that is spatially constrained, and hard-wired. In the MPF all memories are stored in hardwired, conjunctive bindings that encode certain (invariant) features. How can such an architecture represent tree structures of arbitrary depth, and how can it process novel sentences whose specific structures are not recognized by the existing hard-coded classifiers (i.e., how can it explain the *productive* use of language)?

The answer to this question will be gradually developed in the upcoming sections, but in short it goes as follows: an MPF-like architecture may serve as the basis of a neural grammar, but there still need to be some special operations that can combine primitive elements from that grammar into sentences. This is the familiar *binding problem* of language [e.g., Jackendoff, 2002]. Specifically, I propose that the MPF needs to be extended by a mechanism for *dynamic binding* of its nodes into more complex representations; moreover, for learning and processing of novel sentences the network must be able to bind remote nodes that have no pre-existing (hard-wired) relationship. The following sections discuss some recent developments in the research on dynamic binding in vision, which invite an analogy with dynamic binding in language processing (section 2.3).

Second, even though the MPF is modeled after the visual system, the framework (as well as its Bayesian instantiation [George and Hawkins, 2009]) ignores the possible functional role of topology. In section 2.6.1 I propose that, by analogy to vision, the cortex exploits topology for the purpose of encoding and updating graded syntactic category membership.

A third problem is that the MPF offers no account of how event-like (personal) memories, such as a recent visit to a museum, are stored in the cortex. Apparently the MPF describes a *semantic* memory store, but not an *episodic* memory (for definitions of episodic and semantic memory see section 2.7). Yet, episodic memories are an integral part of our memory system, and the episodic memory of large, reusable parts of sentence analyses presumably plays an important role in language learning and processing. A realistic account of language processing and acquisition would therefore benefit from the presence of an episodic memory system, through which the analyses of sentences that an individual has processed in the past are made accessible, such that they can be recruited for the analysis of novel sentences. The role of episodic memory in (exemplar-based) language processing will be discussed in section 2.7, and a computational model hereof will be presented in Chapter 6.

2.3 Neuro-biological solutions for the binding problem

In vision, the binding problem deals with the question how the brain succeeds at integrating multiple features, such as color, shape and motion, which are processed in separate channels, into a single percept of an object [see e.g., Zeki, 1993]. In particular, when multiple objects are present in a scene it is not a trivial task to attribute each feature to the correct object. For instance, in an image with a red square and a green triangle, red must be bound to the square, and not to the triangle. Binding is also necessary to integrate the different parts of an object that is extended in space, and distinguish it from other objects with which it may overlap (e.g., crossing curves) — the latter task is known as *contour binding*.

Within the visual processing pathway there exist cells that are dedicated to binding specific configurations of visual features, such as edges, simple shapes, or even the familiar face of your grandmother (so-called grandmother cells). Features that are very common benefit from being hardwired (conjunctively coded) in dedicated brain cells, and as a consequence they can be processed fast, automatically and in parallel during feed forward processing (that is, propagation of activation in the bottom-up direction). This is called *static*, or *conjunctive* binding. Yet, even with the aid of a vast number of such specialized cells the brain cannot in general solve the binding problem for vision, because there can be infinitely many possible configurations of visual features, for instance in irregular shapes. Although the feed forward sweep can successfully group many hardwired feature constellations, for most complex tasks (e.g., texture segregation, curve tracing, and visual search) more flexible forms of binding are necessary to group the features of an object. The situation is comparable to the language domain, as there are not enough neurons in the brain to bind every unique sentence [e.g.,

van der Velde et al., 2004].

There are at least two competing neural theories that explain *dynamic binding* in vision. Perhaps the most popular theory, *synchronous neuronal firing*, has been proposed by von der Malsburg [1981] as a way to bind features over long distances. Although synchronous firing is a ubiquitous and well-studied phenomenon [see e.g., Singer and Gray, 1995], the theory does not clarify how two cells that fire in synchrony actually know of each other, and how the bound percept is made explicit as a single entity. This would require dedicated 'synchrony detectors', and thereby the responsibility for solving the binding problem is shifted to the latter. There have been several studies that show that no direct relationship exists between synchrony and perceptual grouping [e.g., Lamme and Spekreijse, 1998, Roelfsema et al., 2004], and many researchers believe that synchronous firing is perhaps only an epiphenomenon of binding [Palanca and DeAngelis, 2005]. Moreover, synchronous firing would run into trouble as a theoretical explanation when bindings have a partial temporal overlap, as is the case for sentence processing and for most other realistic stimuli.

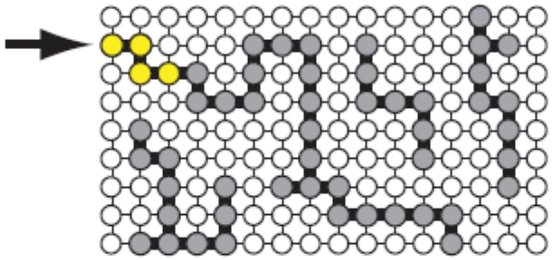


Figure 2.5: The contour binding algorithm. Serial spread of a label (light color) through bottom-up enabled neurons (dark gray). Reproduced from [Roelfsema and Spekreijse, 2005]

An alternative, and in my opinion preferable theory asserts that dynamic binding is a serial process mediated by the spread of an enhanced neural firing rate, which is thought to be the neural correlate of attention [Roelfsema, 2006]. This works in two stages, as is illustrated for the case of contour binding in Figure 2.5:

1. During a feed forward sweep neurons are activated that are tuned to specific conjunctions of features in their receptive fields (i.e., the static bindings). This 'enables' the neurons for subsequent dynamic binding.

2. Dynamic binding occurs when a label (i.e., attention) spreads through recurrent connections between enabled neurons, that are topological neighbors in feature space. (The fact that neighboring neurons respond to similar features (e.g., orientation) forces the binding process to follow a continuous contour line.)

Empirical evidence for a serial, attention-mediated process in contour binding is described in Roelfsema and Spekreijse [2005], where electro-physiological recordings were used to measure response times of individual neurons in the monkey brain.

I propose that a similar serial binding mechanism as used in contour binding is also fundamental for productive language use. As a corollary recurrent serial binding gives rise to recursive phenomena in language, as discussed in the next section.

2.4 Recursion, substitution and dynamic binding

A view that has recently gained some popularity among generative linguists is that the only thing special about human language is recursion [Hauser et al., 2002]. Recursion can be defined with respect to a formal (context free) competence grammar. It means that a rewrite rule can directly or indirectly rewrite to its own left hand side, thereby allowing arbitrarily many applications of the same rewrite rule in a derivation. This definition assumes, of course, globally defined and discrete syntactic categories.

Not only is the view that recursion is universal across the world's languages contested [e.g., Evans and Levinson, 2009, Everett, 2005], careful reflection reveals that the definition of recursion as an empirical phenomenon is rather troublesome. According to the formal definition the recursivity of a language depends on the choice of syntactic categories that describe the grammar of the language. However, this choice is usually a matter of linguistic convention, as is the granularity of the categorization. For instance, the *verb* category can be subcategorized according to the type of arguments it selects for, or according to tense and aspect.[4] Of course, whether a grammar is recursive depends on the degree of granularity of its categories. In linguistic theories such as Radical Construction Grammar [Croft, 2001] category labels are locally defined around specific constructions. From their point of view recursion is entirely absent. Clearly then, there cannot be a definition of recursion that is independent of any grammatical representation or theory; recursion derives from the assumption that syntactic categories have global meanings, and is a property, or rather an artifact, of the formal model of grammar, and not of the real world.

Yet, nobody would dispute that there is an *empirical* phenomenon in language related to recursion, that allows language users to produce and understand compound, and hierarchically nested propositions in discourse, and this phenomenon

[4]Some state-of-the-art probabilistic parsers apply a technique — called state-splitting — that splits categories into more fine-grained categories, and thereby improve their accuracy significantly [Petrov et al., 2006].

deserves an explanation. Language users who judge *Mary likes chocolate* to be a grammatical sentence of English also accept *My little brother likes chocolate*, indicating that their grammar allows for *substitution* of an item for another item of the same kind (i.e., substitution class) in a sentence or in discourse (see sections 4.5.1 – 4.5.2 for an elaborate discussion of systematicity in language). In formal linguistics substitution refers to an operation of combining rewrite rules based on the identity of syntactic category labels. Obviously in a symbolic approach to language substitution is not an issue, because it is implicitly assumed that variables are global and that their content can be replaced globally. Within a biologically realistic connectionist framework however, which lacks the notion of variables, substitution is not trivial to represent.

One of the central claims of this thesis is that *the neural correlate of substitution is attention-mediated serial binding*. Hence a uniform cortical mechanism underlies binding both in vision and in language. Contour binding is only one example of the general solution of the brain, across all modalities, to dealing with complex or compound stimuli which involve *recurrent* processing.

This challenges the position of Hauser et al. [2002], who believe that recursion is central to human language processing. Evidently, recursion is only an epiphenomenon of a more basic cognitive ability, namely substitution — once the mind acquires an ability for doing substitution recursion comes for free. As such, recursion is not special in any sense for language, and neither is it a special quality that sets humans apart from other primates.

2.5 Connectionist implementation of serial, dynamic binding

In order to implement serial binding, or substitution, in a connectionist model of language processing one must assume localist representations, and the existence of *complex* network units.In that case substitution can be construed by analogy to serial contour binding [Roelfsema, 2006] with only a few adaptations. While serial contour binding is realized by passing a label between neighboring nodes in the visual topology, substitution is realized by passing a 'tag' (containing information about the sentence position) between units in a 'syntactic' topology. When an identical tag is present in two bound units, of which one is a complex unit, this implements a 'pointer' from (a substructure of) the complex unit to the bound unit (details in section 5.7). This allows the complex unit to delegate responsibility for processing a part of its input stimulus (e.g., a phrase) to the unit that is pointed at, hence the replacement of one tag by another inside (a substructure of) a complex unit amounts to substitution.

In case the brain invokes a serial binding mechanism for language processing, the neural assemblies further need to have a capacity to retain the bindings in working memory for a short while. The assumption that neural assemblies

are endowed with small short-term memories is in fact not so far fetched. Recent findings in neuro-biology indicate that working memory is locally sustained by short-term plasticity, through calcium-mediated synaptic facilitation in the recurrent connections of neocortical networks [Mongillo et al., 2008, Barak and Tsodyks, 2007]. Residual calcium is thought to act as a buffer that maintains the presynaptic connections in working memory for a time on the order of one second after the synaptic activity has terminated.

Although the idea of dynamic binding has not yet entered the mainstream connectionist modeling literature, its implementation, involving short-term neuronal memories, is not contrary to the spirit of connectionism. The local short-term memories can for instance be realized in a neural network by providing for an additional set of fast changing weights, that transiently store pointers to bound nodes. Hummel and Biederman [1992] implement a dynamic binding algorithm, which is very similar in spirit to the algorithm proposed by Roelfsema [2006], in a connectionist network for complex shape recognition. The challenge for dynamic binding is that temporarily bound neurons must be 'tagged' by a tag that identifies their membership of the bound group. Hummel and Biederman [1992] solve this by spreading the tag via an extra set of Fast Enabling Links (FELs) between neurons, that operate on a much faster time scale than, and independently of the regular feed forward connections.

Another connectionist model that reserves a central role for substitution between nodes (or dynamic binding) is the STORM model [McQueen, 2005, McQueen et al., 2005]. The STORM model is a localist connectionist network, built on top of a Kohonen Self-Organizing Map (see the next section). It implements a special mode of interaction between network units (the so-called 'override' operation), in which connectivity depends only on the topological location of the active units, and is insensitive to their actual activation. The units in STORM have an extra set of weights for remembering context, that is the topological position of units that previously activated them — these 'context weights' act as local short-term memories. In this thesis I will propose yet another solution for dynamic binding for the HPN model (see sections 5.1 and 5.7), but I will borrow the idea from STORM that connectivity between units depends on their relative topological positions.

In sum, I propose that to be able to cope with language processing (and substitution) neural networks need (apart from feed forward connections) a mode of connectivity that has the flexibility to bind any pair of units through serial, dynamic binding, using tags. The capacity for dynamic binding or substitution renders such networks potentially much more powerful than traditional connectionist networks; in particular it allows them to perform *context invariant* operations, which are important to satisfy the *systematicity* of language (see section 4.8.1). All this comes at a price: one has to adjust what for historical reasons has become the standard picture of a neural processing unit. Most importantly, one needs to assume that the units have local memories and an ability to transmit

richer information than just activation levels. It thus makes sense to take as the basic computational units in a neural network for language processing cortical columns rather than neurons [Mountcastle, 1997]. The implementation of these ideas constitutes the major part of this thesis.

2.6 Localist networks and topology

If one assumes that syntactic development involves a process of gradual acquisition of prototypical syntactic categories and their integration within a network topology, then *localist* connectionist networks, such as the Kohonen network [Kohonen, 1998], seem a more logical choice to model this process with than *distributed* connectionist networks (which will be discussed in Chapter 4). The Kohonen network, also called a self-organizing map (SOM), exhibits many of the properties found in the visual system (section 2.1.1), such as localist representation of feature detecting neurons, topological organization and unsupervised, Hebbian learning.

In brief, a Kohonen network consists of an input layer and a layer of 'feature extracting' nodes (the *feature map*), which are placed according to some topological arrangement (usually a rectangular grid). Every feature node in the feature map is fully connected with every node of the input. When an input is presented to the network, the node that receives most activation becomes the 'winner', and the activation of all other nodes is inhibited. The winner is the node whose weights have the minimal distance to the input.

Subsequently, the weight of the winning node is updated by decreasing the difference between the weight and the input, using a Hebbian learning algorithm. This reinforces the winning node, which makes it more likely to classify similar inputs in subsequent presentations - the so-called winner-takes-it-all effect. An important feature of Kohonen networks is that they organize the topology by updating not only the weights of the winner, but also of nodes surrounding the winning node on the grid. As a result an area in the neighborhood of the winner will respond to similar input vectors, and the topology of the map self-organizes.

2.6.1 The role of topology in a neural theory of syntax

Given the success of Kohonen networks in explaining the formation of the visual topology [e.g., Miikkulainen et al., 1997], can they also be used to study the formation of a syntactic topology? Several models exist that investigate the exploitation of topology for syntax, often by extending the SOM with a capacity for recurrent processing [e.g., Voegtlin, 2002, Mayberry III and Miikkulainen, 1999, McQueen et al., 2005]. Since recurrent networks are a main topic in Chapter 4 I will not discuss these models here.

It is worth mentioning a study by Ritter and Kohonen [1989] that investigated

whether it is possible to induce 'word categories' from distributional information. Ritter and Kohonen [1989] trained a SOM on three-word-sentences generated by an templates made of word categories. The input to the network consisted of precomputed vectors representing words together with their average left and right contexts. Thus, distributional information was explicitly encoded in the input vectors. Their results show that SOMs can learn to represent 'syntactic' word categories as regions in a topological map.

Yet, since this study only involves three-word sentences it does not deal with the question of how the categories induced in the topological map interact, or bind to each other. As in most other applications of Kohonen networks, the formation of a topology is considered a goal by itself, and the topology plays no functional role. Ideally however, one would want the formation of syntactic categories to result from an interaction through the topology (i.e., in a parsing process). This requires a fundamentally different approach to syntax induction with Kohonen networks, some of the challenges of which will be discussed in the next section.

Why would the brain invest such an effort in developing topologies? It is unlikely that topographic order is preserved for aesthetic reasons alone; apparently topology plays a role in determining connectivity patterns. The most plausible answer, I find, is that the topological position is used as an index by higher brain regions to make approximate predictions, exploiting the fact that graded category membership is laid out smoothly along the dimensions of the topology. By virtue of the topological organization of categories the brain's predictions do not have to be exact, which makes them more robust, and a (mis)match between the expected and observed input can be computed based on their topological distance.

From an engineering point of view, relying on topology is a very effective way for the brain to encode relations between categories (or variables), because rather than connecting individual category members (i.e., network nodes) one by one (as is typically done in connectionist semantic networks, [e.g., Collins and Quillian, 1969]) it is sufficient to express a systematic relation only once by pointing to an area, where the category members are clustered. The brain can thus infer the category of a neural assembly from its topological position on the cortical surface, whether in the domain of visual object recognition, motor behavior, or language.

The ubiquity of topological organization everywhere in the brain, and its crucial role in determining connectivity patterns, are strong indications that the brain represent categories locally, and not fully distributed, as is advocated by proponents of distributed connectionism (e.g., the SRN). Topology also seems to fly in the face of the more extreme non-representational theories of connectionism [e.g., van Gelder, 1998], which defend the absence of categories altogether.

2.6.2 Countercurrent systems, top-down and bottom-up networks

One of the difficulties in adopting the Kohonen framework for syntax is that syntax is a 'top-down' phenomenon: syntactic categories are defined by their occurrence in certain sentence contexts, hence their representations should reflect contextual, distributional information. By contrast, in the standard implementation of the Kohonen network the input to the network comes from bottom-up (e.g., through feed forward visual perception). Hence the topology reflects local features implicit in the input vectors and does not incorporate relational information.

Before one can answer the question how a syntactic topology is realized, one must address the question where 'conceptual' (syntactic) categories are located in the brain. An interesting approach to this question is offered by a tradition of cognive modeling known as *countercurrent systems* [e.g., Ullman, 1991, Merker, 2004, Schmidhuber, 1996]. Closely related to the Memory Prediction Framework, it emphasizes that cortical processing proceeds in two relatively independent networks, a *bottom-up network* and a *top-down network*, in which neural activation streams in opposite directions, and which are "seeking to meet at every patch of cortex" [Ullman, 1991]. While the bottom-up network is driven by sensory input and encodes *perceptual* features, the top-down network is task- or goal-driven, and encodes *conceptual* categories and relations.

A similar interaction between two modes of processing is essential in the language domain. Here, phonological, lexical and syntactic categories reside in a goal-driven top-down network, while acoustic categories are sensory driven, hence reside in the bottom-up network. Specifically, syntactic processing takes place in the top-down network. Recently, several publications have appeared advocating the need for some sort of 'countercurrent system' approach to model the interface between neurobiology and linguistics in speech perception [e.g., Poeppel et al., 2008]. According to Poeppel et al. [2008] the fundamental challenge for cognitive science is to find 'linking hypotheses' between acoustic primitives and phonological and lexical primitives.

A strong candidate for a 'linking hypothesis' that has received some empirical support is that cortical columns participate simultaneously in both networks, with one 'pole' activated by bottom-up perceptual stimuli, and the other 'pole' responsive to top-down influences. This is supported for instance by a study involving single-cell recordings in monkeys that have to trace a curve [Roelfsema, 2006]. The study shows that some of the same cells that are involved in feed forward processing enhance their response after some latency (about 200ms), and incorporate information from outside their receptive field, indicating the involvement of recurrent pathways from higher visual areas. Lamme and Roelfsema [2000] review several more studies, using the backward masking paradigm and visual search, that similarly show that in visual processing, after an initial feed forward

sweep (during which perceptual features are detected from bottom-up), additional interaction with top-down processing is needed to invoke visual awareness.

Thus, for simplicity I will assume in this thesis that cortical columns have two poles (a 'conceptual' and a 'perceptual' pole), which locally link the top-down network and the bottom-up network. (Note that in the MPF the function of the conceptual pole is assumed by the 'name' that is passed to the higher level region.) In the auditory / linguistic domain it will be assumed that the same cortical column that encodes the acoustic features of a word also encodes its lexical category (for instance by linking the word for dog to the sound for dog). Most neuro-biological research has focused on topology formation in the bottom-up network, for instance in the visual and auditory cortex, but I argue that a different topology is also realized simultaneously in the top-down network, for instance in syntax.

The idea of using topology to express (syntactic) relations between categories has not, as far as I am aware, been incorporated before in a neural network model of language processing. It will be implemented in the HPN model in combination with serial binding, proposed in section 2.5, to take care of the combinatorial productivity of language. For now, I will leave the question of how this can possibly be implemented to sections 4.8.2 and 5.7.

2.7 The role of episodic memory in language processing and acquisition

An important question that a neural theory of language has to deal with concerns the nature of the smallest productive units of language that are stored in memory. When producing a novel sentence it seems that language users often reuse entire memorized sentence fragments, whose meanings are not predictable from the constituent words. Examples of such *multi-word constructions* are *How do you do?* or *kick the bucket*, but there are also productive constructions with one or more open 'slots', such as *the more you think about* X*, the less you understand*, or completely abstract and unlexicalized constructions. Such multi-word constructions must be memorized by the language learner as a whole, together with their associated meaning, because they cannot be derived compositionally from the rules of a grammar. The third major challenge for a neural theory of language, according to Jackendoff [2002] (see section 1.2), is to explain how such non-compositional multi-word expressions are represented and stored in the brain. In Jackendoff's words: "What aspect of an utterance must be part of long term memory, and what aspects can be constructed online in working memory?" [Jackendoff, 2002, p. 152].

Mainstream generative grammar has argued that constructions exist only in the periphery of language, and that therefore they need not be the focus of a linguistic or learning theory. But, in fact constructions of varying degree of complex-

ity and abstractness are pervasive in language. This is the view of construction grammar [see e.g., Goldberg, 2006], mentioned in section 1.2, and of certain formal theories of syntactic processing such as Data Oriented Parsing (DOP) [Bod, 1998, Scha et al., 1999]. In these theories the assumption is made that constructions (or tree fragments in DOP) have psychological reality as the primitive units of language production, hence they must be stored in their entirety in memory [Scha, 1990]. To account for this fact within a cognitive theory of language it is useful to consider some of the insights from cognitive psychology about the human memory system.

Episodic and semantic memory

In cognitive science a distinction is often made between two memory systems, which are known since Tulving [1972] as semantic and episodic memory.

- Semantic memory is a person's general world knowledge, including language, in the form of concepts (of objects, processes and ideas) that are systematically related to one another.
- Episodic memory is a person's memory of personally experienced events or episodes, embedded in a temporal, spatial and emotional context.

A semantic memory of "bread" *Me lining up in front of the bakery*

Figure 2.6: A semantic memory of bread (left) and an episodic memory of a visit to the bakery (right).

For example, the memory of the walk from your home to the bakery on a rainy Monday morning constitutes an episodic memory, while the concept of bread with all its associations constitutes a semantic memory.[5] Together, episodic and semantic memory constitute our *declarative memory*, which is the memory of events and objects that one can consciously report on.

The distinction between episodic and semantic memory is also important for a cognitive approach to language research, as a substantial part of both our semantic and episodic memory is dedicated to representing linguistic knowledge in

[5]Note that the semantic memory of a concept should not be confused with its 'semantics', which refers to the meaning of the concept.

more or less abstracted form. While abstract rules of grammar are encoded within the semantic memory system, memories of concrete sentences (and sentence fragments) are presumably part of the episodic memory system, since we can often remember them integrally.[6] For this reason it seems a good idea to incorporate the notion of a semantic-episodic memory interaction within a cognitive theory of exemplar-based language processing in order to explain the reuse of sentence fragments in production and comprehension.

Language acquisition

Apart from in production, episodic memory also plays an important role in language acquisition. Much of the knowledge we acquire during our lifetime is gathered through personally experienced events, which are first stored as episodic memories before they are gradually transformed into abstract, semantic knowledge.

The gradual transition from concrete to abstract language use is notable in children's linguistic development, and has been extensively documented [e.g., Tomasello, 2001, 2000a]. As discussed in section 1.2, according to Usage Based Grammar (UBG) children's language moves from a holophrastic stage, during which they memorize and imitate complete utterances, via a stage of item-based speech, during which they start introducing variable content in the slots of constructions (e.g., *I wanna X*), to increasingly abstract and adult-like language.

These studies (and other evidence that I will present in Chapter 8) suggest that language acquisition can be seen as a special case of the process of *memory consolidation*, which describes the storage of concrete episodic memories, and their gradual transformation and assimilation into an abstract, relational semantic memory system through a process of de-contextualization. (Conversely, in the neurosciences recently theories of memory consolidation have started to recognize the relation to grammar learning, and to adopt learning algorithms (such as the inside-outside algorithm) from computational linguistics [e.g., Battaglia and Pennartz, 2011].)

The integration of the idea of an episodic-semantic memory interaction within a neural model of exemplar-based language processing and acquisition will be developed in Chapters 6 to 8. It is based on an original theory about the representation of episodic memories, which will be presented in Chapter 6. As a

[6] Whether the division between episodic and semantic memory applies to linguistic knowledge is not entirely uncontroversial though. An alternative position is that grammars are part of *procedural* memory, since the rules of grammar cannot be consciously reported, while only the mental lexicon is part of declarative memory [Ullman et al., 1997]. In my opinion however, there is a definitional problem surrounding the term declarative memory in cognitive science. I would propose that declarative memory, as opposed to procedural memory, is the evolutionary higher memory system that pertains to the built-in mental model of the world, planning and conscious thought. For humans, language and grammar are an essential part of that memory system.

background the next section briefly surveys the state-of-the-art in neuro-biology on the relation between episodic and semantic memory, and on memory consolidation.

2.7.1 The neurobiology of memory consolidation

The nature of the relation between episodic and semantic memory

Although there is general consensus that episodic memory and semantic memory are not separate modules, but massively intertwined, the exact nature of the relation between episodic memory and semantic memory is at present still an open question. A common view is that episodic memories are constructed as pointers that bind together items stored in semantic memory, both in temporal relations, and in relations between roles or participants in an event [e.g., Shastri, 2002, Eichenbaum, 2004]. Such a conception of episodic memory fits within a popular theory in cognitive neuroscience, the so-called *reinstatement hypothesis of episodic retrieval*, which says that during episodic memory retrieval memory traces (i.e., the pointers) are triggered, thereby reactivating the cortical circuits that were involved in encoding the episodic memory [e.g., Sutherland and McNaughton, 2000, Woodruff et al., 2005]. Further, in order to encode the flow of events associated with a certain experience it is assumed that episodic memories bind static semantic elements into temporal sequences, linked through their shared context [e.g., Eichenbaum, 2004, Levy, 1996] (see Figure 2.7).

The central role of the hippocampus

The hippocampus plays a crucial role as the central brain structure involved in the encoding and subsequent consolidation of episodic memories. It acts as a 'gateway' that automatically registers all attended perceptual stimuli before they are stored as episodic memories [Morris and Frey, 1997]. Lesion studies in human amnesia patients have provided evidence that episodic memories are, at least initially, stored in the hippocampus [e.g. Scoville and Milner, 1957]. After the initial encoding of an episodic memory a very long period of memory consolidation follows (up to several months), during which it has been found that the hippocampus initiates *replay*, mostly during sleep, of the episodic sequences in the neocortex [e.g., Sutherland and McNaughton, 2000] (see [Battaglia et al., 2011] for a good review). It is believed that as a consequence of replay eventually cortical representations of episodes develop that no longer depend on the hippocampus [Squire and Alvarez, 1995].

Storage of episodes in relational networks

Episodic memories are not stored independently of one another, but they are integrated in a functional network around semantic features that they share with

other memories. One of the first indications for the structural organization of episodic memories was the discovery of so-called 'place cells' in the rat hippocampus, which fire every time the rat finds itself in a particular location in a maze [O'Keefe and Dostrovsky, 1971]. O'Keefe and Nadel [1979] have proposed that place cells encode, at the population level, a 'cognitive spatial map' of the environment, that enables the rat to find its way in the maze.

However, since this initial finding many other relational and sequential patterns have also been shown to trigger hippocampal place cells, for instance odor sequences, reward consumption, or any relevant event in a learning task. This has led Eichenbaum et al. [1999], among others, to conclude that hippocampal place cells can encode any general kind of causal or temporal sequence, with spatial relations being only a specific instance of a sequence (of spatial cues).

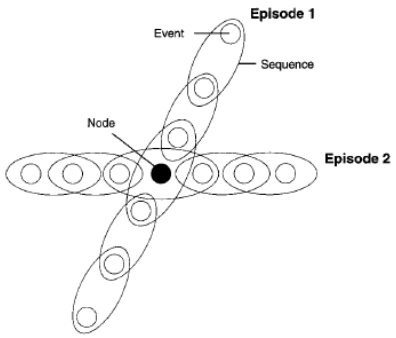


Figure 2.7: Schematic diagram of a relational memory network (reproduced from [Eichenbaum et al., 1999]).

The general picture that has started to emerge is that episodes are stored in the hippocampus in *relational networks* (also called *memory spaces*), where they are linked through shared semantic features [e.g., Eichenbaum et al., 1999, Eichenbaum, 2004] (see Figure 2.7). Episodes chain together sequences of semantic events, and this provides the causal structure in the relational networks. The shared semantic nodes allow the brain to make associative jumps between episodes (i.e., *priming*), while special context-responsive cells preserve the identity of a particular episode across shared elements [e.g., Levy, 1996]. Eichenbaum [2004] proposes that a mechanism of structural integration of episodes, resulting from the detection of their shared structure, is at the core of the consolidation process, leading to de-contextualized representations and strengthened semantic relations between units.

Flexible expression of memories and productivity

A complementary function of the hippocampal relational networks is in mediating flexible and productive use of memory in various cognitive domains, including

language [e.g., Opitz, 2010]. The hippocampus has been implied in problem solving and planning [e.g., Eichenbaum et al., 1990, Eichenbaum and Fortin, 2009]. According to Eichenbaum [2004, p. 110] the idea behind this is that "the elements that encode common features between episodes link memories to one another, allowing one to compare and contrast memories and to make inferences among indirectly related events".

Empirical support for this claim comes from animal studies that show crucial hippocampal involvement in the ability to make transitive inferences, for instance between pairwise ordered odors. Bunsey and Eichenbaum [1996] presented rats with a series of odor pairs with overlapping elements, and trained them to express a preference for one odor in each pair. Their study showed that rats who had their hippocampus removed were not able to infer the underlying hierarchical relations between the pairs (i.e., they could not infer preferences transitively), whereas rats in the control group were able to do so. Apparently the rats had constructed an internal 'relational network' (a hierarchy of pairwise relations), from which they were able to productively infer novel odor associations that they had not encountered before (e.g., $A > B \wedge B > C \Rightarrow A > C$).

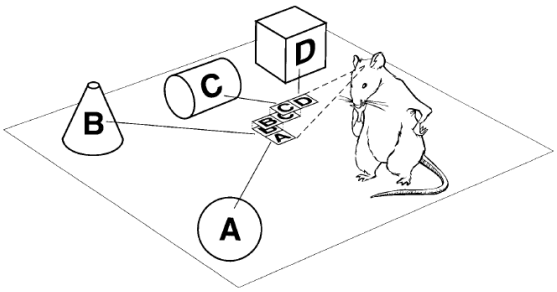


Figure 2.8: The rat's navigation is facilitated by an ability for transitive inference between spatial memories (reproduced from [Eichenbaum et al., 1999]).

In the same vein it is argued that the creation of a relational network of spatial memories enables inferential reasoning about space, and it explains the rat's ability to navigate and find shortcuts in the maze, through transitive inference (see Figure 2.8).

The research in this thesis adopts the view that similar principles as involved in memory consolidation govern grammar acquisition. I propose that in grammar acquisition, like in spatial learning, a relational syntactic map is constructed from concrete linguistic episodes. Like spatial inference, a relational network in the language domain allows for grammatical inference through flexible association of stored linguistic episodes, resulting in productive language use and comprehension of hitherto unseen utterances. The idea of the construction of a relational syntactic network out of linguistic episodes will be given a computational basis in Chapter 8.

2.8 A neural theory of syntax

This chapter offered a quick tour of all the ingredients that I believe are needed to make a neural theory of syntax work. The starting point of the research in this thesis is the assumption that there exists a uniform cortical mechanism that underlies categorization and processing within different modalities, and specifically that syntactic processing can be understood by analogy to visual processing. I argued that the Memory Prediction Framework is a good candidate for such a cortical algorithm, in particular by virtue of hierarchical temporal compression and its proposed mechanism for extracting invariant representations. Yet the MPF is missing some critical features that would allow it to explain the human ability for productive use of language. Such productivity (or combinatoriality) necessitates a dynamic form of binding to account for the syntactic phenomenon of *substitution*. Subsequently, I identified the serial binding process used in the domain of vision as the neural correlate of a substitution operation, and argued that dynamic binding in language presupposes the existence of short-term memories at the neuronal level. Finally I argued for a functional role of topology in determining connectivity patterns between syntactic categories, or substitutability. All this is summarized in a proposal for a neural theory of grammar, which is founded on the following six hypotheses

- There exist cell assemblies in the language area of the cortex that function as neural correlates of graded syntactic categories.
- These 'syntactic' cell assemblies can 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. Paradigmatic relations are expressed by the topology, while syntagmatic relations are realized by temporal compression.
- The neural correlate for substitution is dynamic binding between 'syntactic' assemblies. This is implemented via a serial process of label passing. The dynamic binding process assumes the existence of short-term memories at the level of the syntactic assembly, where labels ('tags') are temporarily stored which serve as pointers to other assemblies. Dynamic binding enables productive use of language as it allows for a substitution operation between assemblies; recursion can be seen as an epiphenomenon of dynamic binding, while the stored tags function as a (distributed) stack.
- A graded measure of substitutability is realized as a distance between neural assemblies within a meaningful topology. Learning a grammar amounts to constructing the topology from examples in a process of self-organization.

- The memory for sentences and other linguistic constructions or idioms is accounted for by an episodic memory system, that stores bound sequences of syntactic assemblies. Language processing can be construed as episodic memory retrieval, while grammar acquisition is an instance of a memory consolidation process from episodic linguistic events to an abstract grammar in the form of a relational syntactic map.

In Chapter 5 I will develop a computational model, the Hierarchical Prediction Network (HPN), that can be seen as one possible instantiation of these hypotheses. The HPN model is still far from a perfect implementation of the neural theory of grammar; for example it does not distinguish more than two cortical levels in the hierarchy, and it cannot deal with non-configurational languages (languages that do not have a fixed word order). Nevertheless, HPN can address some of the most intriguing questions about the neural basis of syntax, which is a major incentive to depart from a formal approach to linguistics (Chapter 1): What kind of brain structures can support the functionality of syntactic categories and rules, considering the productive and systematic nature of language? Where do syntactic categories come from if they are not innate, and how do they acquire their abstract status? How can structural representations of sentences (e.g., parse trees) be efficiently represented in the brain, and how is a parse executed? With the background of this chapter it is in principle already possible to answer some of these questions (they can be found in section 9.3, in reply to Jackendoff's challenges).

2.9 A note on semantics

Before I conclude this chapter, let me clarify an issue that may have been nagging at the reader's mind. Whereas the scope of this thesis is limited to the treatment of syntax, I am well aware that many contemporary cognitively oriented theories of language consider syntax and semantics to be inseparably linked within an individual's representation of linguistic knowledge. Such a view is most notably expressed by Lakoff, who describes the grammar of a language as "a neural system which, far from being an autonomous module, consists of neural connections linking the bodily grounded conceptual and the expressive (phonological) systems of the brain" [Lakoff and Johnson, 1999, p.498].

In construction grammar (CG) [Goldberg, 2003, 2006] the basic units of language are constructions: associations between a semantic frame and a syntactic pattern, for which the meaning or form is not strictly predictable from its constituent parts [see e.g., Kay and Fillmore, 1999]. Despite the focus on syntax in this thesis, it should be noted that the philosophy behind the HPN model, which will be presented in Chapter 5, is compatible with the constructionist (or cognitive linguistic) account of grammar.

Following a common view on semantics held in cognitive linguistics, I consider the term semantics to cover the processed information from the combined non-linguistic modalities (e.g., vision, touch, smell and sound), grounded in perception [e.g., Steels, 1996, 1999], and I do not envisage any formal (e.g., predicate logic) interpretation of semantics. The increasingly invariant and abstract multi-modal categories that are encoded in higher regions of the cortex constitute a semantics, that is an abstract semantic network in the context of which syntactic categories that are derived from linguistic input can be placed. Whereas the framework of the MPF accounts for the extraction of highly abstract visual categories [George and Hawkins, 2009] and can in principle deal with multi-modal categories as well, the HPN model developed in this thesis takes the MPF framework to the field of syntax. Yet, as any other connectionist model, HPN is not designed to be domain specific. The eventual goal is to formulate a neural model that incorporates both abstract multi-modal and linguistic categories, as well as invariant categories that associate (or bind) between form and 'meaning', which linguists refer to as constructions.

In this regard I find the traditional division between syntax and semantics in linguistics rather misguiding. If syntax refers to 'structural regularities' as opposed to arbitrariness in the language domain, then it should be clear that there is also a syntax in the visual domain (for instance, part-whole structure, or structure of spatial relations between objects), as well as in other domains that constitute a semantics. For this reason I preferred to use the term 'structure in language' rather than 'syntax' in the title of this thesis.

A model of syntax acquisition that relies entirely and exclusively on distributional information selects only a small fraction from the abundance of linguistic and non-linguistic input that is available to language learning children, and one should therefore keep in mind that such models do not tell the whole story of syntax acquisition. Apart from sequences of words, there are many other cues that aid children to infer the correct grammatical structure and categories; to name some: joint attention, visual information, prosody (intonation and stress patterns), and prior knowledge of the semantic categories. In particular, the presence of a basic semantic ontology prior to language learning may help to bootstrap syntax, and vice versa exposure to the syntax of adult speakers may help the child to bootstrap and fine tune its ontology [Steels, 1999, Borensztajn, 2006a]. There exist computational models of construction grammar that assume or construct a complete ontology prior to grammar acquisition [e.g., Chang, 2001, Steels, 1997, 2004].

Even though the focus of this thesis will be exclusively on syntax, I do assume that the principles (of learning and parsing) that are proposed here have universal applicability, both for syntactic categories and for categories from other domains. Indeed, HPN was originally conceived as a generic neural network model of category learning, while keeping in mind that the problem of categorization is treated uniformly in the brain, irrespective of the modality of the input stimulus, whether

it is language or vision.

Chapter 3

Symbolic approaches to language processing

This chapter presents a brief overview of formal grammars, parsing and statistical natural language processing (NLP). In the symbolic paradigm, as opposed to the connectionist paradigm, syntax is viewed as dealing with symbol manipulation, and therefore it can be cast which facilitates its formalization within a rigid mathematical framework. The material covered in this chapter can be found in many textbooks on (statistical) natural language processing, hence it may be skipped by the reader who has a (computational) linguistic background (except perhaps section 3.1.10 and the final section). I will introduce context free grammars (CFG), probabilistic grammars, and different techniques for parsing with CFGs, among which chart parsing. A special focus is put on left corner parsing, with its application to the HPN model in mind. Then I spend a few sections discussing variations of the PCFG statistical model, particularly Data Oriented Parsing. Furthermore, two techniques for unsupervised grammar induction will pass in review: the Inside Outside algorithm and Bayesian Model Merging (BMM). I will end the chapter with a discussion of the problems of the symbolic approach to syntax, focusing on gradedness and prototypicality of syntactic categories, and the acquisition process. I will consider evidence from the language acquisition and construction grammar literatures that motivates a connectionist approach to language acquisition.

3.1 Introduction to formal syntax and parsing techniques

Syntax is the branch of linguistics that studies the rules and constraints governing the construction of sentences in natural languages. In the formal approach to linguistics, the syntax of a language is strictly separated from its semantics, and deals only with surface form. The rules of a language are captured in a **formal grammar**, which is a mathematical device that allows for generating all and only sentences of a certain language.

In most natural languages sentences are internally organized in phrases, which group together words that syntactically behave as a unit. Such units, also called **constituents**, can be identified because they are not broken apart when they move to a different syntactic position within the sentence. Grammars make use of abstract **syntactic categories**, which are classes of words or phrases that show similar syntactic behavior, that is they can occur at the same syntactic positions, and have the same expansion possibilities. The main criterion for syntactic category membership is substitutability: if a word (or phrase) can be replaced in a sentence by another word or phrase while preserving the grammaticality of the sentence then both belong to the same category [Harris, 1951]. Nouns, verbs and adjectives are examples of syntactic word categories, also known as *parts of speech*. Linguists also distinguish phrasal categories such as the Noun Phrase (NP), which is a phrase headed by a noun (e.g., *the fat lady*)), the Verb Phrase (VP), (e.g., *watches television*) and the Prepositional Phrase (PP) (e.g., *on the sofa*).

The syntactic analysis of a sentence into phrases tells us how to determine the meaning of a sentence from the meaning of the words: for languages with a fixed word order (e.g., English), the phrasal analysis can be used to identify the **grammatical relations** in a sentence, such as the subject, the predicate and the object (who does what to whom). There are two kinds of structural relations: horizontal between sisters in a constituent (linear precedence), and vertical between parent and daughter (immediate dominance). It is common practice to represent the phrase structure analysis of a sentence as a **phrase structure tree**. The leaf nodes of the tree are called **terminals**, while the internal nodes are called **nonterminals**.

3.1.1 Rewriting grammars

A **rewriting grammar** consists of a set of **rewrite rules** (also called *productions*), which are applied to the syntactic categories. An example of a rewriting grammar is given in Table 3.1.

Formally, a rewriting grammar is a tuple $< V_N, V_T, S, \mathcal{R} >$ with V_N a set of nonterminals, V_T a set of terminals, S the start nonterminal, and $\mathcal{R}$ a set of

nonterminals: S, NP, VP, DET, N, V
terminals: *the, woman, read, the, book*

LHS		RHS	P(RHS—LHS)
S	→	NP VP	1.0
NP	→	DET N	1.0
VP	→	V NP	1.0
V	→	*read*	1.0
DET	→	*the*	1.0
N	→	*woman*	0.7
N	→	*book*	0.3

Table 3.1: Example of a probabilistic context-free grammar

rewrite rules.

In a **context free grammar** (CFG), a subclass of the family of **phrase structure grammars**, the possibilities for expanding a rewrite rule depend solely on the label of the nonterminal on the left hand side, and not on any surrounding context. Context free rewrite rules are of the form $A \rightarrow \alpha$, where A is a nonterminal ($A \in V_N$) and α is any sequence of terminals and nonterminals ($\alpha \in (V_N \cup V_T)^+$).

Using the rewrite rules of a formal grammar one can generate all possible sentences of the corresponding formal language. Conversely, one can find a derivation for every grammatical sentence. The **left-most derivation** of a sentence is a sequence of context free rules that derives the sentence, starting with the start symbol (S), and followed by repeated replacement of the left-most non-terminal by an appropriate rewrite rule until only terminals remain in the resulting string. Table 3.2 shows an example of a derivation (but not left-most) of the sentence *The woman read the book*, using the context free rewriting grammar of Table 3.1. In a CFG every derivation maps uniquely to exactly one parse tree. The parse

S	
→	NP VP
→	DET N VP
→	DET N V NP
→	DET N V DET N
→	*The woman read the book*

Table 3.2: Derivation of the sentence *The woman read the book*

tree corresponding to the derivation above is given in Figure 3.1.

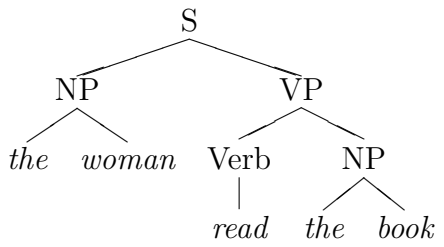


Figure 3.1: Example of a parse tree

3.1.2 Automata and the Chomsky hierarchy

Context free grammars (CFG) are but one of a number of families of rewriting systems, each of which is characterized by certain restrictions on the forms of rewrite rules allowed. **Regular grammars** (RG) are more restrictive than context free grammars, because their rewrite rules can have at most a single nonterminal on their right hand side. In a regular grammar each rule is of the form $A \rightarrow w$, or either $A \rightarrow w\ B$ (generating only right-branching trees), or $A \rightarrow B\ w$ (generating only left-branching trees). Here A and B are nonterminals, and w is a terminal. One says that RGs have weaker **generative power** than CFGs, because the class of languages that they can generate is a subset of the class of languages that can be generated with CFGs.

Context sensitive grammars, on the other hand, have stronger generative power than CFGs. Their rewrite rules are of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where α, β and γ are strings of terminals and/or non-terminals.

A different point of view on a grammar is as an abstract computing device that can recognize or generate a language, a so-called **automaton**. This perspective is illuminating if one is interested in the comparison between symbolic language devices and connectionist models of language, which will be the subject of the next chapter. Automata have a virtual tape for reading input strings and writing output strings, and the more powerful automata have an optional extra 'scratch tape' that can be used for working memory. A finite state automaton has a finite number of internal **states**, including special initial and final states. Depending on the input and the state it is in, the automaton decides what action to perform: read, write, and/or move the tape, or for certain classes of automata read and write to memory. An automaton recognizes a string $w_1, \ldots, w_n$ to be in the specified language if, starting from the initial state, and after reading all words w_i from the tape in order and performing the prescribed actions, it is in the special end state.

Rewrite systems can be arranged in a hierarchy of increasing generative power, which is called the Chomsky hierarchy [Chomsky, 1957] (see Table 3.3). For each rewrite system in the Chomsky hierarchy there exists a corresponding automaton that recognizes exactly the same family of languages.

Language	Automaton	Rewrite rules
Regular	Finite state automaton	$A \rightarrow w$, either $A \rightarrow w\ B$, or $A \rightarrow B\ w$
Context free	Pushdown automaton	$A \rightarrow \gamma$
Context sensitive	Linearly bounded Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$, $\gamma \neq \epsilon$
Recursively enumerable	Turing machine	$\alpha \rightarrow \beta,\ \alpha \neq \epsilon$

Table 3.3: The Chomsky hierarchy. Uppercase letters denote non-terminals, Greek letters denote strings of terminals and/or non-terminals, w denotes a terminal, ϵ is the empty string.

A question of ultimate import is where natural languages are situated within the Chomsky hierarchy. In a famous result in formal linguistics Chomsky [1957] shows that natural language is at least context free. The argument rests on an understanding that a generative model of language needs to capture at least the human capacity for producing recursive structure. Language speakers can nest constructions, such as *if . . . then*, that correspond to pairwise dependencies inside each other, in principle arbitrarily many times, to produce recursive sentences like the following (reproduced from [Chomsky and Miller, 1963]): *Anyone*$_1$ *who feels that if*$_2$ *so-many*$_3$ *more*$_4$ *students*$_5$ *whom we*$_6$ *haven't*$_6$ *actually admitted are*$_5$ *sitting in on the course than*$_4$ *ones we have that*$_3$ *the room had to be changed, then*$_2$ *probably auditors will have to be excluded, is*$_1$ *likely to agree that the curriculum needs revision* (the subscripts indicate dependencies between words).

This alleged ability for recursion to arbitrary depth can be invoked, together with the pumping lemma for finite state automata (FSA), to prove that English is richer than a finite state automaton language.[1] Because the FSA generates infinite strings by means of simple iterations, an FSA can by no means produce an infinite, center embedded string. However, natural languages can 'in competence' express such center-embedded forms, hence it follows that the human language system requires at least the computational power of context free grammars.[2] Nowadays it is believed that the grammars of natural languages are situated somewhere between context free and context sensitive (i.e., they are mildly context sensitive). In the Swiss German language, as well as in Dutch,

[1]The proof uses the fact that sentences like the example in the text are of the form $\{XX^R | X \in \{a, b\}^*\}$ (R indicates reversal), which can be shown to be non-regular by first intersecting it with the regular language aa^*bbaa^* to yield $\{a^n b^2 a^n | n \geq 1\}$, and then applying the pumping lemma [Partee et al., 1990]. Informally, the pumping lemma for FSA expresses that strings of arbitrary length can only be produced from a finite FSA if the grammar has a simple loop or iteration and pumps it. If the length of the string exceeds the number of states, which is finite, this means that a single state has been used more than once, hence there is a loop.

[2]Obviously, Chomsky does not assume a human ability to produce unbounded strings; nevertheless for the proof the limit case needs to be invoked as a device to distinguish the behavior of families of grammars, otherwise any finite output of a context free grammar could be approximated by a very large FSA.

there exist certain productive constructions (called cross-bracketing) that require stronger generative power than that of context free grammars. Evidence that Swiss German is not context free has been provided by Shieber [1985]. In section 4.6 I will argue that the Chomsky hierarchy also plays a pivotal role in the discussion about whether certain types of connectionist networks are fit to characterize human language.

3.1.3 Parsing

Parsing is the process of finding the derivation(s) by which a given sentence might have been generated. The resulting phrase structure tree that is constructed from parsing a sentence is called a **parse**. A **parser** is a computer program that implements a search for a parse, given a sentence and a grammar.

There are different parser search strategies, each with its advantages and disadvantages. A discussion of these can be found in many text books [e.g., Jurafsky and Martin, 2009, Chapter 13], but I will mention them briefly here.

- A **top down parser** starts with the start symbol S and searches rules whose left hand side non-terminal matches the left-most non-terminal in the derivation.

- A **bottom-up parser** starts with the words of the sentence and matches rules to their right hand side until it reaches the root of the tree.

- A **left corner parser** [Rosenkrantz and Lewis II, 1970, Demers, 1977] combines top-down and bottom-up parsing, and proceeds incrementally from left to right. This makes left corning parsing a cognitively quite plausible parsing strategy. The 'left corner' is the left-most symbol on the right hand side of a phrase structure rule. A left corner parser has three operations: *shift*, *attach* and *project*. One starts by putting the *goal category* S in the *goal stack*, and shifting to the first word of the sentence, which becomes the *left corner category*. Given a goal category, which is predicted from top-down, and a 'left corner category', which is derived from bottom-up, one can either attach the left corner category to the goal category if they are the same, or project the left corner category if it matches the left corner of a rule in the grammar. After a projection the remaining categories on the right hand side of the rule become goal categories, and one recursively does left corner parsing of each. If there is no left corner category (because it has been attached), one shifts to the next word of the sentence and makes it the left corner category. The process is schematically illustrated in Figure 3.2.

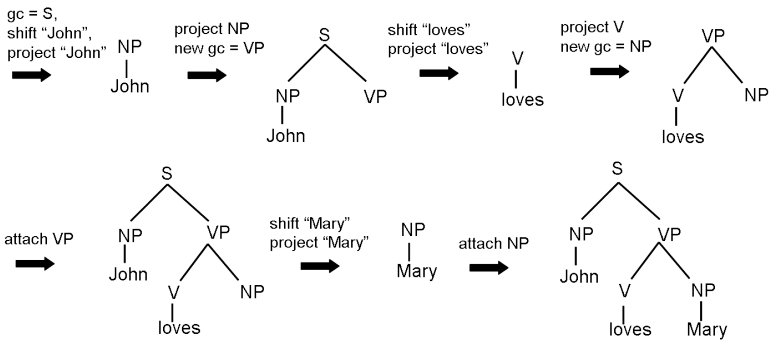


Figure 3.2: Left corner derivation of *John loves Mary.* gc is short for goal category.

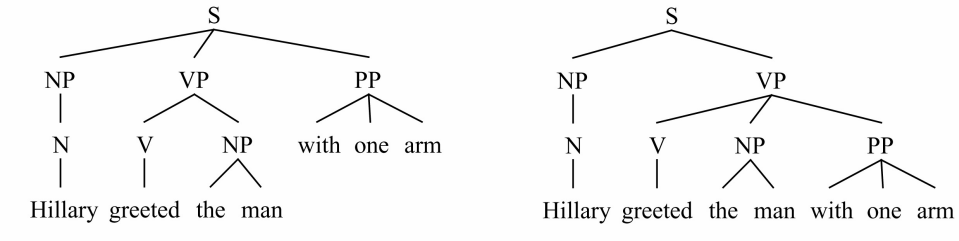


Figure 3.3: Attachment ambiguity

3.1.4 Probabilistic context free grammars

In general, a single sentence can have a large number of different parses, giving rise to **structural ambiguity**, for instance regarding the attachment of a preprositional phrase in the parse tree (see Figure 3.3). The number of possible typically parses grows exponentially with the length of the sentence. It is possible to disambiguate between the parses by ranking them according to their probability.

To assign a probability to a parse one assumes that the steps in its derivation, i.e., the applications of rewrite rules, are independent of each other. In that case the event of a parse can be broken down into independent simpler events, for which the probabilities can be estimated from a corpus. A context free grammar can be upgraded to a probabilistic CFG (PCFG) by enriching the context free productions with probabilities (see the example in Figure 3.1). In context free grammars one makes the **independence assumption** that the choice of the expansion of a rewrite rule is conditionally independent given its left hand side (LHS)

$$\forall A \in V_N : \sum_{\alpha:\ A \to \alpha \in \mathcal{R}} P(\alpha|A) = 1 \tag{3.1}$$

where A is a nonterminal, α is the right hand side of a rewrite rule, and $\mathcal{R}$ is the

set of rewrite rules in the grammar. Using the context freeness assumption and the chain rule, it follows that the probability of a derivation *der* given sentence X is the product of the probabilities of the rules r_i in the derivation sequence, given their LHS.

$$P(der|X) = \prod_i P(r_i|LHS(r_i)) \tag{3.2}$$

Given a sentence X, and a statistical model G, the most probable parse T is given by

$$argmax_{T \in G(X)} P(T|G) \tag{3.3}$$

where $G(X)$ is the set of parses of X licensed by G.

There also exist probabilistic versions of left corner parsers, which will be discussed shortly.

3.1.5 Treebank estimation

In the statistical parsing world one often wants to estimate the parameters of a probabilistic grammar from realistic data. For this purpose there exist **treebanks**, which are large corpora of sentences annotated with phrase structure trees. One of the most widely used treebanks in statistical NLP is the Penn Treebank, and particularly the Wall Street Journal section [Marcus et al., 1993].

When a treebank is available, the rules and probabilities of a PCFG can be estimated using **maximum likelihood estimation**. A maximum likelihood (ML) estimator finds the parameters of a model M that maximizes the likelihood of the data X, generated by M.

$$M_{ML} = \mathbf{argmax}_M P(X|M) \tag{3.4}$$

Under certain conditions [Prescher, 2003] maximum likelihood estimation is equivalent to **relative frequency estimation**. Let $\mathcal{F}_\mathcal{R}$ denote the frequencies of the rewrite rules in $\mathcal{R}$, then its relative frequency in the treebank is given by

$$rf(A \to \alpha) = \frac{\mathcal{F}_\mathcal{R}(A \to \alpha)}{\sum_{\beta:\ A \to \beta \in \mathcal{R}} \mathcal{F}_\mathcal{R}(A \to \beta)} \tag{3.5}$$

The denominator denotes the total count of rules with LHS equal to A.

3.1.6 Parser evaluation

Natural language parsing systems are usually evaluated on their performance by comparing the computed parse trees on a test set with a so-called gold standard, which are the manually annotated parse trees from the same test set. The constituents of the parse trees are compared one by one by their span and label, using the standardized *PARSEVAL* metric [Manning and Schütze, 2000, p.

432]. In this metric constituents with the S label and constituents representing preterminal nodes are omitted from the evaluation.

Let $\mathcal{C}(T_C^i) = \{T_C^1 \dots T_C^n\}$ be the constituents of the correct parse and $\mathcal{C}(T_G^i) = \{T_G^1 \dots T_G^m\}$ the constituents generated by the parser. Two measures are generally used to capture the success of the parse:

- **Labeled Recall** (**LR**) is the number of matching constituents in the parse relative to the total number of constituents in the correct parse (the proportion of correct constituents that are recognized by the parser)

$$LR = \frac{\sum_i |\mathcal{C}(T_C^i) \cap \mathcal{C}(T_G^i)|}{\sum_i |\mathcal{C}(T_C^i)|}$$

- **Labeled Precision** (**LP**) is the proportion of correct constituents relative to the total number of constituents in the produced parse.

$$LP = \frac{\sum_i |\mathcal{C}(T_C^i) \cap \mathcal{C}(T_G^i)|}{\sum_i |\mathcal{C}(T_G^i)|}$$

The F-score is defined as the harmonic mean of LP and LR:

$$F = \frac{2 \times LP \times LR}{LP + LR}.$$

3.1.7 Probabilistic left corner parsing

Manning and Carpenter [1997] propose a probabilistic extension of a serial left corner parser, which conditions the probabilities of parser moves on the left corner (lc) and goal category (gc), rather than conditioning on the left hand side of a production, as does a PCFG. Left corner probabilities are *not* context free; they depend on the history of the parser moves and hence they can capture the fact that expansion probabilities of non-terminals typically depend on their position in the tree. For instance, in the WSJ treebank a noun phrase (NP) expands nine times more often to a personal pronoun in subject position than in object position.

In the model of Manning and Carpenter [1997] parsing is a two-step generative process: given a left corner (lc) and a goal category (gc), one must first decide whether to attach (if lc=gc) or project a new rule, hence

$$P(attach|lc, gc) + P(project|lc, gc) = 1$$

Then, in case of project, one must decide to which rule to project, hence

$$P(project|lc, gc) = \sum_{rules\ with\ lc} P_{project}(rule|lc, gc)$$

Assuming the sentence S is given, and the parsers moves are conditionally independent given the left corner and goal category, then the probability of a left corner derivation der_{lc} is

$$\begin{aligned} P(der_{lc}|S,G) &= \prod_{attachments} P(attach|lc,gc) \\ &\times \prod_{projections}(1 - P(attach|lc,gc)) \times P_{project}(r|lc,gc) \end{aligned} \tag{3.6}$$

If one is interested in the string probabilities (e.g., for a language model, or to compute prefix probabilities) one must also consider the probability of shifting to a left corner

$$\sum_{lc} P_{shift}(lc|gc) = 1$$

and include the shift decisions in the calculation of the joint probability of the derivation and the sentence. The base probabilities can be estimated from a treebank, as every parse tree in the tree bank corresponds to a unique left corner derivation. One must first construct the left corner derivation, and then count the frequencies of attachments and projections, conditioned on a left corner and goal category.In section 7.1.2 I will describe a left corner language model that includes shift probabilities, and develop a chart parser that can efficiently compute prefix probabilities.

3.1.8 Variations to the PCFG statistical model

In practice the standard treebank PCFG model has not been very successful at parsing natural language, because the independence assumptions it makes are too strong. While the important role of context has long been accepted for processing in other cognitive domains, the importance of context in sentence processing has only recently started to be appreciated (partly due to the predominant role of generative grammar in linguistic theory). With the context freeness assumption of the PCFG two relatively independent sources of contextual information for disambiguating between parses are ignored: **lexical context**, which captures the dependency on previous words in the sentence, and **structural context**, which captures the dependency on the relative position in a parse tree.

The absence of lexical dependency information in a PCFG makes it a poor **language model**, in the sense that it is bad at predicting word transitions, with even a simple trigram model outperforming the PCFG. In the PCFG the probability of expanding a verb phrase either as $VP \rightarrow V\ NP$, or as $VP \rightarrow V\ NP\ NP$ depends only on the VP label. Yet, clearly this assumption is wrong, since transitive verbs are more likely to expand to $VP \rightarrow V\ NP\ NP$ than intransitive verbs. One way of coping with the problem is by **lexicalizing** the grammar. Assuming that lexical dependencies are mostly carried between the head words of phrases and their dependents, one may enrich the constituent labels in the treebank trees with their head words, which are percolated up in the tree, and subsequently

estimate the PCFG from the lexicalized trees [see e.g., Collins, 2003, Charniak, 2000].

The other weakness of the standard PCFG is the assumption of independence of structural context. As was shown in section 3.1.7 for the case of the NP constituent, in treebank PCFGs the possibility of expanding a certain constituent depends to a large extent on its position in the tree. Again, the structural independence assumption of PCFGs can be relaxed by incorporating structural information in the node labels. Johnson [1998] showed that the parsing accuracy of the treebank PCFG is greatly increased by relabeling the nodes with information concerning their ancestors. In the simplest case, using parent annotation, the labels are enriched by adding @P, where P is the parent label. As a more general approach, **history based parsing** exploits the idea that the parser moves are conditioned on n previous parser decisions in the derivation history. This, too, can be implemented by relabeling the tree nodes to include extra ancestral information. Left corner parsing is an instance of history based parsing because, as discussed in section 3.1.7, the parser moves are conditioned on two (or more) points in the derivational history rather than on structure.

A general scheme for optimizing the performance of treebank PCFGs, without relying on lexical information, was proposed by Klein and Manning [2003], using vertical and horizontal Markovization. Vertical Markovization (v) refers to the use of parent annotation that captures the vertical history of a node (i.e., its vertical ancestors), while horizontal Markovization (h) refers to a binarization process of local trees (i.e., CFG productions) with more than two children, splitting off sister nodes from the head child in leftward and rightward direction. The $v \leq 2,\ h \leq 2$ Markovization serves as the basis for a variety of improved parsing models with increasingly refined labels, with state-of-the-art results obtained by the state-splitting model of Petrov and Klein [2007].

3.1.9 Data Oriented Parsing

The idea behind Data Oriented Parsing (DOP) [Bod, 2003, Scha et al., 1999] is that the primitive productive units of language are tree fragments of arbitrary size and depth, which can be (partly) lexicalized or abstract. The cognitively motivated assumption of DOP is that a person's linguistic knowledge consists of parts of utterances which are stored in memory, and reused when parsing novel sentences. DOP is formalized as a Tree Substitution Grammar (TSG) enriched with probabilities. Tree Substitution Grammars form a generalization over context-free grammars and a subclass of the Tree Adjoining Grammars [Joshi, 2004]. In DOP, the elementary tree fragments of the TSG can in principle be any subtree occurring in a treebank of syntactically annotated sentences, provided every local tree is included in the subtree together with all its daughters. Two elementary tree fragments can be combined by means of the substitution operator $\circ$ if the left-most non-terminal leaf node of the first fragment is identical to the root node

of the second fragment (see Figure 3.4). A derivation of a sentence in DOP is a sequence of elementary tree fragments $t_1 \circ t_2 \circ \ldots \circ t_n$ such that the root of the first fragment is of the special category S and the leaves of the resulting tree are the terminals that constitute the sentence .

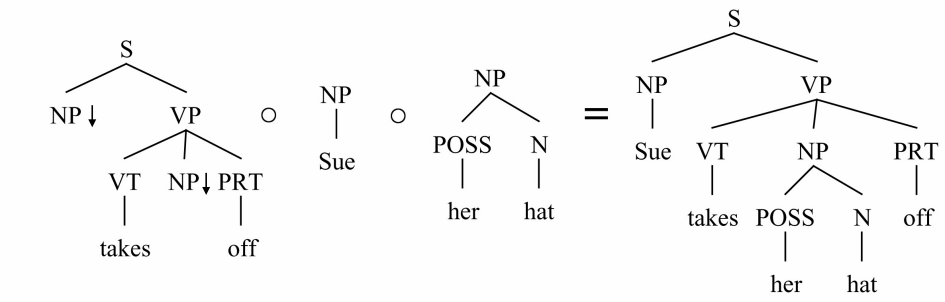


Figure 3.4: Derivation for *Sue takes her hat off*. Substitution sites are marked with ↓.

It is assumed that the probability of any substitution is context independent. Hence, the probability of a derivation can be written as the product of probabilities of the subtrees used in the derivation, conditioned on their roots:

$$P(t_1 \circ t_2 \circ \ldots \circ t_n) = \prod_i P(t_i|R(t_i))$$

Here $P(t_i|R(t_i))$ is the probability of a subtree t_i, whose root $R(t_i)$ matches the label at the substitution site. Whereas in a PCFG a parse corresponds to a unique derivation, in DOP there can be many different derivations (with different substitution sites) associated with a single parse. As a consequence in DOP the probability of a parse is not equal to the probability of a derivation, but to the sum of all derivations that give rise to the same parse tree.

Several alternative **estimators** for finding the probabilities $P(t_i|R(t_i))$ of the fragments (the parameters of the DOP grammar) have been proposed. The earliest estimator is known as DOP1 Bod [1998], and assigns probabilities to elementary tree fragments based on relative frequencies conditioned on the root $R(t)$ of the tree.

$$P(t|R(t)) = \frac{|t|}{\sum_{t':R(t')=R(t)} |t'|} \tag{3.7}$$

The more recent 'push-'n-pull' estimator [Zuidema, 2006, 2007] is based on the idea that probability mass is pushed or pulled from subtrees in the treebank based on the discrepancy between their observed frequency and their expected frequency.

3.1.10 Children's grammars grow more abstract with age

Borensztajn et al. [2009a] used Data Oriented Parsing as a statistical approach to automatically discover the most probable multi-word constructions in children's utterances. The study was conducted on the Brown corpus [Brown, 1973] from the CHILDES database [MacWhinney, 2000]. This corpus contains transcribed longitudinal recordings of three children, Adam, Eve and Sarah, enriched with syntactic annotation [Sagae et al., 2007]. Each of the children's subcorpora was split into three parts of roughly equal size, representing three consecutive time periods.

Assuming the children's underlying grammar is a Tree Substitution Grammar, the parameters of the grammar were estimated using the push-'n-pull estimator [Zuidema, 2006, 2007], after which standard statistical parsing techniques were used to find the most probable derivation of any sentence in a corpus. This yielded a decomposition of the sentence into those elementary tree fragments that together constitute a hypothesis on how the sentence was generated. This way, the most probable constructions used by each child were found.

#	Period 1	#	Period 2	#	Period 3
33	where N go	9	you V it	10	you V it
11	I V it	8	I do **NEG** want INF **X**	5	you **X** and PRO **X**
6	what that N doing	7	I V it **JCT**	5	will you V it
5	take N off	6	you V it	4	can PRO put **X**
4	who N that	6	where PRO went	4	a ADJ one
4	do **NEG** V it	6	let me V it	4	do **NEG** know PRO:WH PRO V
4	have N on	5	I can **NEG** V it	4	do **NEG** know PRO:WH PRO:DEM V
4	you V it	5	going INF make DET N	4	and PRO:WH is that
3	what N doing	5	let us play DET game	4	can PRO put **OBJ LOC**
3	put N on	5	what kind N that	4	what is PRO:DEM for
3	put **OBJ** on	4	going put **OBJ** in it	4	I can not V it
3	do not V me	4	a N cake	3	how **AUX** you V PRO:DEM
3	take **OBJ** off	4	I V him	3	maybe PRO is **X**
3	where N N go	4	in DET kitchen	3	you V this ADV
3	I V some	4	you V me **COMP**	3	I going **X** off

Table 3.4: Adam's most frequent discontiguous multi-word constructions (shown are only the leaf nodes). (**X** indicates a dummy node in a parse tree, introduced by the conversion from dependency to binary constituent annotation.)

Table 3.4 shows Adam's 15 most frequently used discontiguous constructions (i.e., constructions in which the lexical items are separated by variable slots) of each period. In the figure, part of speech tags are indicated in capital, and grammatical relations appear in bold capital. The found constructions cover many interesting linguistic phenomena from the language acquisition literature, such as the progressive, use of auxiliaries, clausal constructions with *want* and *think* (which appear among the most frequent *contiguous* constructions), and particle verbs (*take **OBJ** off*, *going put **OBJ** in it*).

From the Table it may be noted that, whereas in Period 1 most constructions are very concrete, starting from Period 2 constructions become more abstract (as can be seen from the increased number of substitution sites). This tendency is also apparent from the increased use of variable pronomina (e.g., PRO and PRO:WH).

In a subsequent quantitative analysis, features of the used elementary trees, such as the number of nodes, non-terminals and terminals and the depth of a construction, were averaged and compared across the periods for every child separately. The possibility of an artifact due to increasing average sentence length with age was ruled out by comparing only constructions from sentences of the same length across periods. While no effect of age was found for construction size, depth or number of nodes, a strong age related effect was found in all children for the average *abstraction* of used constructions, showing that abstraction of children's constructions increases with age. Abstraction was defined as the ratio between non-terminals (i.e., variable slots) and terminals (lexical items) in a construction.

These results constitute evidence against the so-called *continuity assumtion* of linguistic development [e.g., Crain and Thornton, 2005], according to which children have (in competence) the same syntactic categories and rules as adults. The observed progressive abstraction supports instead a *usage based* view of grammar acquisition [e.g., Tomasello, 2005], which predicts a gradual development from a concrete, item-based to an abstract grammar.

3.1.11 Chart parsing

Chart parsing is an efficient way of parsing, because it stores intermediate parse results in a chart, avoiding the redundancy caused by reanalyzing parts of the sentence every time the parser backtracks. Chart parsers instantiate a technique known as **dynamic programming**. One instance of a chart parser, the Earley parser [Earley, 1970], is particularly interesting because it computes parses from left to right. I will cover the Earley parser and its probabilistic extension in some detail, because it serves as a framework for a probabilistic left corner chart parser that I will develop in section 7.1.2.

The Earley chart parser

The Earley parser keeps track of all derivations that are consistent with the input string moving from left to right through the input. When it arrives at position i in the sentence it performs some operations (see below) that generate new *states*, and stores the states pertaining to position i in cell i of the chart.

A **state** describes a particular instantiation of a production in a pending derivation. Given a rewrite rule $X \rightarrow \alpha$, a state is of the form $i : {}_kX \rightarrow \lambda \bullet \mu$ (also called a **dotted rule**), where the dot separates the processed part of the

right hand side of the production (i.e., the part that has been accounted for by the input string up to position i) from the unprocessed part.[3] The number i is the current position in the input string (i.e., $x_0 \dots x_{i-1}$ has been processed); k is the position in the input string where X started expanding. A state is called **complete** if the dot is on the right of the entire RHS.

The Earley parser starts a derivation with initial state

$$0 : \ {}_0 \ \rightarrow \bullet \ S$$

and uses three operations that move the derivation forward by adding states, depending on previous states in the chart and on the current input. For each input symbol X_i and corresponding chart cell i the Earley parser performs all three operations exhaustively, until no new states are generated, before moving to the next input symbol (adapted from [Stolcke, 1995]).

- **Prediction**
  For each state $i : \ {}_kX \rightarrow \lambda \bullet Y \ \mu$ in chart cell i
  where Y is a non-terminal anywhere in the RHS,
  and for all rules $Y \rightarrow \nu$ expanding Y, add states

  $$i : \ {}_iY \rightarrow \bullet \ \nu$$

  Each prediction corresponds to a potential expansion of a non-terminal in the derivation.

- **Scanning**
  For each state $i : \ {}_kX \rightarrow \lambda \bullet a \ \mu$ in chart cell i
  where a is a terminal symbol that matches the current input X_i, add the state
  $i + 1 : \ {}_kX \rightarrow \lambda \ a \bullet \mu$
  A state produced by scanning is called a *scanned state*.

- **Completion**
  For each complete state $i : \ {}_jY \rightarrow \nu \bullet$ in chart cell i
  and each state in chart cell $j, j \leq i$, that has Y to the right of the dot,

  $$j : \ {}_kX \rightarrow \lambda \bullet Y \ \mu$$

  add the state

  $$i : \ {}_kX \rightarrow \lambda \ Y \bullet \ \mu$$

  A state produced by completion is called a *completed state* (which is not the same as a *complete state*).

[3] Notational conventions are adopted from [Stolcke, 1995].

A derivation of sentence $x_0 \dots x_{l-1}$ is successful if after processing the last symbol the parser is in the final state

$$l :\ _0 \ \rightarrow S \bullet$$

where l is the length of the input string.

An efficient probabilistic Earley parser that computes prefix probabilities

Stolcke [1995] shows how the Earley parser can be upgraded to a probabilistic chart parser, which apart from keeping track of the most probable derivation, also computes string probabilities incrementally, using dynamic programming. These are marginalized probabilities over all derivations that produce a certain (sub)string. An important reason for computing string probabilities from a PCFG is that they permit a comparison of the performance of the PCFG with language models that do not produce a syntactic analysis (e.g., Markov models, or the Simple Recurrent Network). For several tasks, for instance probabilistic prediction of the next word in a sentence, it is useful to know word transition probabilities. In a PCFG the word transition probability $P(x_{i+1}|x_0 \dots x_i)$ can be computed given the string probabilities of two prefixes, $x_0 \dots x_i$ and $x_0 \dots x_{i+1}$, since $P(x_{i+1}|x_0 \dots x_i) = P(x_0 \dots x_i x_{i+1})/P(x_0 \dots x_i)$. Formally, the prefix probability is the sum of all sentence probabilities having $\mathbf{x}$ as prefix

$$P(S \stackrel{*}{\Rightarrow}_L \mathbf{x}) = \sum_{\mathbf{y} \in \Sigma^*} P(S \stackrel{*}{\Rightarrow} \mathbf{x}\ \ \mathbf{y})$$

Given a PCFG G, a non-terminal X, and a string $\mathbf{x} = x_0 \dots x_i$ over the alphabet Σ of G, the following string probabilities are defined (adapted from [Stolcke, 1995])

String probability The string probability $P(X \stackrel{*}{\Rightarrow} \mathbf{x})$ is the sum of the probabilities of all left-most derivations $X \Rightarrow \dots \Rightarrow \mathbf{x}$ producing $\mathbf{x}$ from X.

Forward probability The forward probability $\alpha_i({}_kX \rightarrow \lambda \bullet \mu)$ is the sum of the probabilities of all constrained paths of length i that end in state $i :\ {}_kX \rightarrow \lambda \bullet \mu$

Inner probability The inner probability $\gamma_i({}_kX \rightarrow \lambda \bullet \mu)$ is the sum of the probabilities of all paths of length $i - k$ that start in state $k :\ {}_kX \rightarrow \bullet\ \lambda\ \mu$ and end in $i :\ {}_kX \rightarrow \lambda \bullet \mu$, and generate the input symbols $x_k \dots x_{i-1}$.

Stolcke [1995] shows that the prefix probability of prefix $x_0 \dots x_i$ can be computed by summing their forward probabilities over all scanned states.[4]

$$P(S \stackrel{*}{\Rightarrow}_L \mathbf{x}) = \sum_{l:\ {}_kX \to \lambda x_i \bullet \mu} \alpha_l({}_kX \to \lambda x_i \bullet \mu)$$

In [Stolcke, 1995] equations are given for the incremental updates of the inner, the forward and the Viterbi probability, using dynamic programming.

The algorithm also computes the so-called **Viterbi parse**, which is the most probable parse of the string. It can be updated dynamically while building the chart by keeping track in each state of the maximal path probability leading to it, as well as the predecessor states associated with that maximum probability path (i.e., the **Viterbi predecessor states**). Once the final state is reached, the maximum probability (Viterbi) parse can be recovered by tracing back the path of Viterbi predecessor states.

3.2 Unsupervised grammar induction with context free grammars

One of the arguments for doing statistical natural language processing (SNLP) is that, like connectionist networks, statistical models use frequencies to account for the gradual change of grammars that happens during language acquisition [Manning, 2003]. Although most SNLP models make no claims of cognitive reality, in recent years much progress has been made with the task of unsupervised grammar induction from realistic data [Klein and Manning, 2002, Bod, 2007]. In this section I will introduce two different statistical techniques for unsupervised grammar induction from unlabeled text, based on PCFGs. The first, the Inside-Outside algorithm, assumes that at least the structure (i.e., the rules) of a grammar is known, and performs a parameter search to estimate the rule probabilities. The second, Bayesian Model Merging, assumes that initially every sentence is associated with a unique rule, and performs a structure search to find a more compact grammar, meanwhile updating the rule probabilities.

3.2.1 The Expectation Maximization Algorithm and the Inside Outside Algorithm

As was discussed in section 3.1.5, if a language learner would have access to an annotated treebank, then the rules and probabilities of a PCFG can be estimated from it using relative frequency estimation. However, in real life the sentences one

[4] It is sufficient to sum over the scanned states alone, because they represent the beginnings of all derivations of complete sentences that start with the prefix. Every such partial derivation ending in a scanned state can be completed with probability one.

hears are not accompanied by a syntactic analysis, hence the parameters of the model that one wants to estimate cannot be observed directly; they are *hidden*. Still, even if the derivations are hidden it is often possible to estimate production probabilities from a corpus of plain text, if one knows at least the rewrite rules of the grammar that generated the sentences beforehand. In that case one can resort to a technique called *Expectation Maximization* (EM).

The idea behind the EM algorithm is that the parameters of the model are initialized to arbitrary values, then the expected values of the hidden variables are calculated based on the initial model parameters. Then the maximum likelihood hypothesis given these values can be computed, by replacing the hidden variables by their expected values, resulting in adjustment of the model parameters. This process is then iterated, such that eventually the likelihood converges to a local maximum.

- In the E-step the expected values of hidden variables are calculated, given the approximated parameters.
- In the M-step, the model parameters are set to values that maximize the data likelihood, given the expected values of the hidden variables.

The instantiation of EM for estimating the parameters of a PCFG is called the Inside-Outside algorithm [Lari and Young, 1990]. Here the parameters are the rule probabilities, and the hidden variables are the derivations (the corpus is unlabeled).

In the the E-step of the Inside-Outside algorithm, the expected counts of the rules are computed from the derivations, given the samples and the current rule probabilities. In the M-step the relative frequency of the rules is computed by equation 3.5.

Clearly, the assumption of EM that the structure of the grammar is known in advance is an oversimplification of the task facing language learning children, who must somehow also figure out the rules of the grammar from plain text (assuming these are not given innately). If a grammar must be learned from scratch without any constraints on the hypothesis space, then the Inside Outside algorithm is not suited as an optimization algorithm anymore, because a maximum likelihood model will always evolve towards a trivial CFG, which has a unique rule for every unique sentence. In that case one needs to define a structure prior over the space of possible grammars, that guides a structure search through that space.

3.2.2 Bayesian Model Merging, Minimum Description Length

As was argued in section 2.2.1, learning syntax can be conceived as a process of discovering differences (or partial matches) by means of alignment between sentences. This idea was formalized by Wolff [1982], who proposed an approach to grammar induction as a structure search through a grammar space by means

of merging and chunking operators. When cast in a Bayesian framework, this approach is called Bayesian Model Merging (BMM) [Stolcke and Omohundro, 1994, Stolcke, 1994]. In BMM the initial rules are set to incorporate all sentences of a given corpus as follows: for each sentence $a_1 a_2 \dots a_l$, new nonterminals $X_1, X_2, \dots, X_l$ are created, as are the following productions:

$$\begin{array}{lll} S & \rightarrow X_1, X_2 \dots X_l & (1) \\ X_1 & \rightarrow a_1 & (1) \\ X_2 & \rightarrow a_2 & (1) \\ \vdots & & \\ X_l & \rightarrow a_l & (1) \end{array} \tag{3.8}$$

The search for symmetries and repeating patterns in the input is aided by two operators, *merge* and *chunk*, which in combination allow to implement a greedy hill climbing search through the space of (context free) grammars by changing existing rules and producing new rules.

- The chunk operator concatenates or chunks repeating sequences, so that the sequence may be stored just once as a unity. It takes a sequence of two nonterminals X_1 and X_2 and creates a new nonterminal Y that expands to $X_1 X_2$. Subsequently all occurrences of the sequence $X_1 X_2$ in the RHS of all productions are substituted by Y.

- The merge operator creates generalizations by forming disjunctive groups (categories) of words or non-terminals that occur in the same contexts. If two nonterminals X_1 and X_2 occur in the same context it replaces them with a single new nonterminal Y. As a result of the replacement two existing productions may become identical and can be stored as a single abstract rule.

At every step of the search, all candidate merges and chunks are considered, and a single one is selected that scores best on a given evaluation function. The evaluation function is chosen to minimize the description length of the model plus the unexplained data.

When compressing data for efficient storage or transmission, usually there is a trade-off between minimizing the size of the theory and minimizing the size of the data that is not explained by the theory. At one extreme, the theory can be compact but too general, so that it does not compress the data much; at the other extreme the theory can be overly specific by having a specific rule for every sentence in the data, but then it grows very large.

Within a Bayesian framework the trade-off can be expressed by constructing an objective function that incorporates the designer's preferences for the model structure, that is a prior probability distribution over a hypothesis space of grammars. One can then search for the hypothesis that maximizes the probability after

some data X is given, the so-called posterior probability. This so-called Maximum a Posteriori (MAP) hypothesis, M_{MAP}, can be expressed in terms of the likelihood and the prior probability by applying Bayes Law:

$$\begin{aligned} M_{MAP} \equiv argmax_M P(M|X) &= argmax_M \frac{P(X|M)\cdot P(M)}{P(X)} \\ &= argmax_M P(X|M) \cdot P(M) \end{aligned} \tag{3.9}$$

The MAP hypothesis takes into account an a priori hypothesis, $P(M)$, called the 'prior', which is a best guess for the model before any data has been observed. This is a probabilistic form of bias. Often the prior is designed such that it biases the model towards simplicity, as a way to implement Occam's Razor, the idea that simpler models are preferred over complex models. If the prior $P(M)$ is uniform, it can be left out of the equation, and in that case the MAP hypothesis is equal to the ML hypothesis.

The maximization of $P(X|M) \cdot P(M)$ is equivalent to minimizing

$$-\log_2 P(M) - \log_2 P(X|M) \tag{3.10}$$

This equation is interpreted in information theory as the principle of *Minimum Description Length* (MDL): for an optimal encoding one should take care that the encoding length of both the theory and the unexplained data is minimized. The Grammar Description Length $GDL = -logP(M)$ is the length needed to encode the model (rounded to an integer number of bits) and the Data Description Length $DDL = -\log_2 P(X|M)$ is the number of bits that is needed to describe the data given the model (assuming an optimal, shared code).

3.3 Limitations of the symbolic approach to syntactic processing and acquisition

In this section I will go a bit deeper into some of the fundamental philosophical and empirical problems with the symbolic approach to processing and acquisition of syntax, in the hopes of clarifying my motivation to pursue a connectionist approach to syntax acquisition in this thesis.

The generative grammar school of linguistics, with its formal treatment of syntax, subscribes to a classical conception of categories as set theoretical entities. Underlying their philosophy is a theory of meaning, that postulates the primacy of objects with fixed properties in the world. This view originates with Frege's interpretation of meaning as *sense* [Frege, 1892], which is an *objective* meaning that is independent of individual minds. In such a view the correspondence between concepts and objects in the real world is fixed a priori. By dealing only with objective, *public* concepts, and ignoring individual minds, it is however difficult for formal theories of syntax to have a convincing account for the acquisition of syntactic categories. The status of categories as set theoretical entities

is a major obstacle for explaining their incremental acquisition, because it implies that a category must exist before it has any members (the so-called *learning paradox*). As a consequence, the classical take on categories has inspired the belief that grammar rules and categories are universally and innately specified by a Universal Grammar [see e.g., Wexler, 1999], and only parameters need to be set for specific languages during learning (the principles and parameters account [Chomsky, 1981]).

In cognitive linguistics, on the other hand, it is commonly accepted that concepts are *mental* constructs, whose function is to build an internal model representing the external world, in service of the survival of the individual [see e.g., Crick and Koch, 1998]. Categories and concepts are believed to be *prototypical*. Prototypical categories are characterized by family resemblance, typicality and similarity effects [Rosch and Mervis, 1975, Rosch, 1978], and incrementally constructed by induction from examples, rather than innate.

3.3.1 Syntactic categories are prototypical and graded

There are some strong arguments in favor of prototypical and graded syntactic categories, and against the idea of universal syntactic categories. A major issue concerns the circularity in the methodology used by linguists to establish the major syntactic categories. Syntactic category membership is established using a method — so-called distributional analysis — that refers to the roles that a category typically fills in certain constructions. The problem of this method is that for the task of identifying syntactic categories often the 'core' English constructions are selected, such as the predicate construction, but categories that appear in other, less common or foreign language constructions are ignored. Croft [2001] argues that this is a circular definition of syntactic categories, because the defining constructions can be selected conveniently, such that they fit existing preconceptions about universal categories.

From inspecting some atypical linguistic constructions it is apparent that there are many exceptions to the major syntactic categories. For example, a typical feature of nouns is that they can occur in a *pluralization* construction, as in *Betty and Sue stubbed their toes*. Yet, ** Betty and Sue lost their ways* is not a grammatical sentence. Another test for nouniness is *pronominalization*, as in *Sam held his breath, and then released it*. Yet, ** Harry took his time, but wasted it* is agrammatical. Furthermore the *gapping* construction, as in *I lost my breath, and she hers* is not a reliable test for nouniness either, because ** I took my time and she hers* is not grammatical (the examples are from Lakoff [1987]).

Cognitive linguists such as Lakoff [see e.g., Lakoff, 1987] take the fact that for all these defining constructions counterexamples can be found as evidence for the prototype-based nature of syntactic categories. Not only nouns, but just about every syntactic category in the English language has been shown to exhibit prototype effects [Ross, 1967], such as typicality (some words are more typical of

a category than others), similarity (not all words within a category are equally interchangeable), and vague boundaries.

Certain words defy categorization into one of the standard linguistic categories altogether. An example from Manning and Schütze [2000, p. 12] is the word *nearer*, as it occurs in

(3.11) We live *nearer* the water than you thought.

In this example nearer simultaneously shows properties of an adjective (the fact that near is used in the comparative form), and of a preposition (only prepositions, but not adjectives, can occur in front of a direct object). Thus, nearer is somewhere in between an adjective and a preposition, suggesting that these categories are not discrete but graded.

A further argument against the universal nature of categories is the fact that the granularity of categories is arbitrary, since it depends on linguistic convention alone according to which criteria words are classified into syntactic categories; for instance, verbs can be subcategorized according to the type of arguments they select for (e.g., transitive and intransitive verbs), or according to tense and aspect (e.g., gerund, past participle). In radical construction grammar (RCG) [Croft, 2001] such considerations are driven to their logical endpoint. RCG holds that language consists of a large inventory of constructions (a so-called *constructicon*), and syntactic categories are defined locally with respect to the constructions in which they occur. Thus, constructions are the primary objects of linguistic analysis, while lexical and phrasal category labels are *emergent*.

Finally, there are also ontogenetic arguments for the graded nature of syntactic categories. From a wealth of empirical studies in language development we know that children do not use the same syntactic categories as adults [Tomasello, 2000a, van Kampen, 2003]. Tomasello's work on so-called 'verb islands' Tomasello [2001, 2000b] shows that in early child language verb constructions are developed in isolation, like small islands, and every verb seems to have its own inflections. Apparently, inflectional rules are not initially generalized over all verbs, but used locally for particular verbs. Studies in this tradition describe how stage after stage apparent 'rules' enter the child's speech, and how categories and rules are gradually fine tuned through abstraction over the input, and become more adult-like. Such a *usage based* view of grammar learning is consistent with a prototypical and graded account of categories.

A similar case can be made based on studies of historical language change and grammaticalization. An all-or-nothing categorical view of language cannot account for the gradual shifts in use and meaning of words and expressions that evolve through time. Manning [2003] gives many examples, among which are the gradual evolution of the expression *kind of*, as in *kind of cute dress*, from a noun into an adverbial expression, the evolution of *going to* into an auxiliary marker of future tense, etc. Such gradual phenomena can only be modeled if one assumes a statistical or prototypical view of language.

3.3.2 Limitations of the probabilistic approach to syntax

Although one may think that the objections raised against the generative 'principles and parameters' approach to grammar acquisition do not apply to the probabilistic paradigm, this is not entirely true. Some of the assumptions about the nature of categories have been carried over to the field of statistical NLP. Although the probabilities themselves are continuous, still in most work in statistical NLP probabilities are put over discrete syntactic categories, such as NPs and VPs, familiar from traditional linguistics. Recently, the parsing world has seen some progress with models that split nonterminals into progressively finer categories [e.g., Petrov et al., 2006]. This suggests that it might be useful "to explore modeling words [and phrases] as moving in a continuous space of syntactic category, with dense groupings corresponding to traditional parts of speech [and phrasal categories]" [Manning, 2003] (words in brackets added by me).

Incorporating topological constraints for learning

Another assumption borrowed from traditional linguistics concerns the global scope of syntactic variables, as implied by the independence assumptions of the PCFG. Grammar induction algorithms based on the PCFG, such as Bayesian Model Merging (BMM) and Inside-Outside (IO) make use of this assumption when they make global updates to all occurrences of a certain variable in the annotated corpus, appealing to some kind of central control mechanism. For instance, in BMM, following a merge operation the label of a variable is replaced globally in the entire corpus, and in IO the parameters of the grammar are updated globally in the maximization step. In light of the empirical studies of grammatical development in children mentioned above, which point at a Usage Based Grammar (UBG) [e.g., Tomasello, 2005], these are not very plausible assumptions.

As explained above, in the usage based view children's learning process follows an incremental trajectory from simple constructions to complex and abstract constructions. Thereby, linguistic categories begin with local scope (i.e., item-based), and the scope gradually expands as categories become more abstract and system-wide.

Radical Construction Grammar (RCG) similarly assumes that the scope of syntactic categories is limited to the specific constructions they participate in. Learning a grammmar in RCG is conceived as the construction of a structured network of constructions, the 'constructicon', by linking constructions through shared categories. In the process the scope of the categories is locally updated, hence widens only gradually.

To account for the observed gradual expansion of the scope of syntactic variables (in children's linguistic development) seems to require a modeling approach that makes use of a network topology, on top of which local interactions between

constructions can be defined. In this view the constructicon is seen as a structured network, and increasing abstraction of the categories is a by-product of the self-organization of the network. Note that such an approach to modeling grammar acquisition leads to a much more complex and interactive learning dynamics than an approach based on the PCFG, because the independence assumptions no longer hold.

While the above considerations provide linguistic and psychological arguments, there is also an important neuro-biological consideration to opt for a neural network approach. If one aims to model language acquisition as it is implemented in the brain then this imposes constraints on the possible learning algorithms: in a neurally plausible model only operations are allowed that can be executed through local interactions (see in this respect the discussion in section 4.2.1).

In sum, there are several good reasons for taking a connectionist approach to syntax acquisition. Besides the prospect of modeling the development of a *constructicon* within a network architecture, and exploiting the possibility of a learning dynamics within a topological space of graded syntactic categories, there is also a constraint from the locality of learning. The connectionist approach will be explored in the next chapter.

Chapter 4

Connectionist approaches to language processing

In this chapter I will discuss connectionist models of language processing in the context of the systematicity debate. *This debate is so called because opponents of connectionism argue that one of the most essential features characterizing natural languages is its* systematicity, *and that connectionist models of language processing cannot explain this. I will argue that the systematicity debate suffers from the lack of a clear operational definition of systematicity, and I will propose a set of criteria for the systematicity of language that is stricter than what is usually accepted [Hadley, 1994]. I will show that despite the fact that it has been discounted by connectionists, the Chomsky hierarchy is still relevant for the systematicity debate; specifically, it gives minimum requirements for the structure bias that a systematic model of language must possess. I will argue that the Simple Recurrent Network does not satisfy the proposed set of criteria for systematicity, and I will critically review the claims of proponents of distributed connectionism that the SRN does implicitly encode knowledge about word categories and constituency. Finally, I will sketch an alternative, localist approach to connectionism, using dynamic binding and complex units, that does meet the systematicity requirements.*

4.1 Introduction

According to a definition by one of its founding fathers,

> connectionism is an approach to modeling cognition based on the idea that the knowledge underlying cognitive activity is stored in the connections among neurons. In connectionist models, knowledge is acquired by using an experience-driven connection adjustment rule to alter the strengths of connections among neuron-like processing units.

[McClelland et al., 2010]

The best way to understand the connectionist paradigm is to place it in opposition to the symbolic paradigm, according to which cognition involves the manipulation of symbols according to rules, like the rules of arithmetic. The connectionist view of computation is inspired by our knowledge of the nervous system, and focuses on the role of activation spreading among local processing units, or neurons. I will assume the reader has some background in connectionism, but to refresh I will recapitulate some basic terminology, without any pretense of completeness.

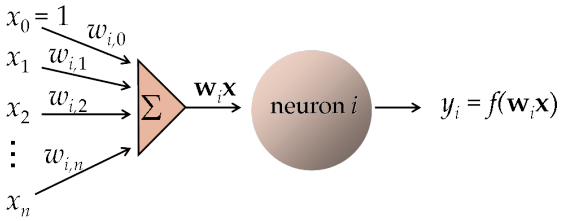


Figure 4.1: The McCulloch & Pitts neuron (image design: Stefan Frank).

The simplest model of a neuron is the McCulloch-Pitts neuron [McCulloch and Pitts, 1943] (see Figure 4.1). Neuron i receives input from n other neurons with activations $x_1, \ldots, x_n$, to which it is connected in a network. Each connection has a *weight* $w_{i,j}$. The total input s_i to neuron i is the weighted sum of the inputs plus a bias term θ_i

$$s_i = \sum_j w_{i,j} x_j + \theta_i \tag{4.1}$$

The neuron's output activation y_i is a function $\mathcal{F}$ (the activation function) of its total input: $y_i = \mathcal{F}(s_i)$. This activation function is usually chosen to be either the step function, a linear function, or a sigmoid (S-shaped) function, i.e. $\mathcal{F}(s_i) = \frac{1}{1+e^{-s_i}}$.

An *artificial neural network* consists of multiple artificial neurons that are connected in a certain network architecture, or topology. Usually one designates special input neurons and output neurons; the other neurons are called *hidden*

neurons. In a *multi-layer perceptron*, also known as a *feedforward network*, neurons are organized in layers, with full connectivity only between neurons in adjacent layers (see Figure 4.2). It has been shown [e.g., Hornik et al., 1989] that a multi-layer perceptron with only one layer consisting of an unrestricted number of hidden units suffices to approximate any function to arbitrary precision, provided the activation functions of the hidden units are non-linear (the universal approximation theorem).

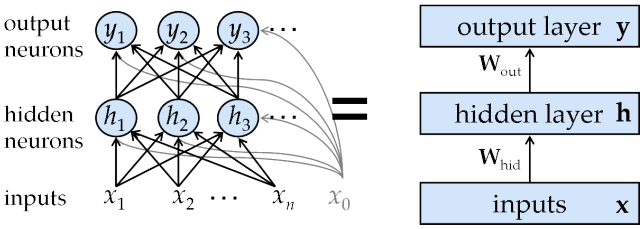


Figure 4.2: A two-layer feedforward network (image design: Stefan Frank).

Neural networks learn by adjusting their weights after processing an example. In supervised learning a *target* t_j is provided to which the network's output is compared, allowing to calculate an error E. An example of a supervised learning rule is the *Delta-rule*

$$\Delta w_{i,j} = \partial E / \partial w_{i,j} = \gamma \cdot (t_j - y_j)\ x_i \tag{4.2}$$

where γ is the learning rate. There exists a wide variety of connectionist models, which are applied to various cognitive tasks, including language, semantics, visual processing, cognitive development, and more. Implementation details depend on the designer's assumptions about functional connectivity in the brain, and the designer's wishes and goals regarding the function of the model. Due to space limitations I cannot provide a survey of the field, but see [e.g., Bishop, 1996, Kröse and van der Smagt, 1996].

4.2 The systematicity debate — critique of connectionism

A computational model of human cognition, including language and reasoning, must explain the fact that people's knowledge seems to be organized around systematic facts. This is evidenced by our *productive* use of language : our ability to produce and understand innumerably many sentences that we have never heard before, on the basis of only a finite number of words, suggests that there is some combinatorial system in place that allows us to construct a novel sentence using some kind of linguistic rules. For instance, it has been argued that

the formation of the regular past tense in English by adding '-ed' to the stem of the verb is governed by some kind of productive rule, otherwise people would have to store every past form besides the present form in a memory for lexical forms, and would not be able to generalize to novel verbs.

The observed systematicity of language has been capitalized upon in a famous argument by Fodor and Pylyshyn [1988], which says that a minimum requirement for a device (such as the brain) to deal with language processing is an ability for symbol manipulation. They claim that the brain must be able to represent universal rules and variables, as in mathematical equations. They further argue that since connectionist models cannot represent variables they are unsuited for symbol manipulation, which they see as a requirement for dealing with the systematic relations in language. Rather, for language processing one needs symbolic devices (of which a computer is a typical example), that can perform operations over symbols, or variables.

The standard connectionist response to the critique of [Fodor and Pylyshyn, 1988] is that even though the emergent linguistic behavior appears on the surface to be combinatorial and rule-like, from this it cannot be inferred that the underlying neural representations are combinatorially structured [e.g., Rumelhart and McClelland, 1986]. Rather than internalizing explicit representations of linguistic categories and rules, it is sufficient for the cognitive system to have implicit knowledge of how to interact with the external symbol system that characterizes the superficial form of language. The battle over the status of explicit mental rules has been waged for a large part around the English past tense, starting with the proposal of Rumelhart and McClelland [1986] for a distributed neural network of past tense acquisition (and subsequent proposals by among others Plunkett and Marchman [1996]), whose model in turn was criticized by Pinker and Prince [1988] and Marcus [1995] for its lack of systematicity.

Another area that has taken a prominent place in the systematicity debate is syntax. Sentences in a natural language like English display an organization into word groups, so-called *constituents* or *phrases*, which behave as functional units that can engage in structure dependent relationships. To process such syntactic phenomena it seems that our cognitive system must have at least an ability to utilize structural representations, if not build explicit internal representations of compositional structure.

There exists a long list of phenomena in language that seem to rely on hierarchical constituent structure. For instance, number agreement between a subject noun and a verb must be maintained even if they are separated by one or more intermediate clauses (so-called long distance dependencies). Other examples are *auxiliary fronting* and *wh-inversion* [e.g., Chomsky, 1972, Crain, 1991]. While few cognitive scientists dispute that constituent structure plays an important role in language, there is disagreement among researchers of competing schools about how a model of the cognitive system can account for apparent structural dependencies. The claim of Fodor and Pylyshyn is that connectionist models

cannot produce constituent related phenomena in language because they lack a mechanism for combining simple representations into more complex, structured representations.

Following Bechtel and Abrahamsen [2002] one may distinguish two major connectionist responses to the structure encoding challenge. One connectionist strategy has been to design connectionist networks that are specially designed to deal with the explicit representation of compositional structure. Examples of this approach are RAAM [Pollack, 1988] and Smolensky's Integrated Connectionist/Symbolic Cognitive Architecture (ICS) [Smolensky and Legendre, 2006], which will be discussed in the next section.

Another connectionist response has been to deny the need for explicit mental representations of compositional structure. This is the leading view among proponents of distributed connectionism, advocating the use of recurrent networks like the SRN ([Elman, 1991]; see section 4.4.1). They believe that the apparent superficial constituent structure observed in language can be dealt with without relying on internal representations of combinatorial structure, assuming that recurrent networks somehow implicitly encode the knowledge for working with external structural representations. I will come back to this response in section 4.4.2.

Although I will not support the claim of Fodor and Pylyshyn [1988] that only symbolic devices can exhibit the systematicity needed for modeling language, I will argue in the upcoming sections that both connectionist replies, representing two extreme positions, are untenable. I will then develop an alternative explanation for connectionist systematicity, which shows how certain notions of 'graded categories' and flexible, learnable 'rules' can be assimilated within a connectionist architecture.

4.2.1 Why connectionism?

Fodor and Pylyshyn's critique raises the question: what is gained by a connectionist approach to cognition and language? Although connectionist models are obviously designed to mimic properties of the brain, there exists a fair amount of skepticism, especially among computational linguists, about whether connectionist models give a better insight than symbolic or statistical models into how the brain deals with cognitive tasks, such as language processing. Neural networks are often described as 'black boxes', whose mathematical properties are not well-understood. In order to motivate the connectionist approach it is necessary to pursue a more concise characterization of the objectives of the connectionist agenda than is usually provided in the textbooks.

In text books connectionist systems are typically characterized based on their possession of certain architectural features, such as spreading activation, locality of processing, and so on, while in my view the crucial difference between the connectionist and symbolic approach to cognition concerns the issue of learning

extensional meaning: Whereas in *symbolic* systems the meanings of the operational units, the variables, are fixed and pregiven, and therefore their extension is *global*, or system-wide, the extensional meaning of the primitive units in a connectionist system is locally constructed, and must be learned. A variable in a symbolic system is a *symbol* in the sense of Peirce [1903]: a token with an arbitrary relation between form and meaning. Assuming the brain encodes meaning in connections between the units of a neural network, then it is hard to see how it can be assigned globally or innately, because meaning does not come out of nowhere: what a neuron represents has to be constructed from experience, through interaction with other neurons and the environment; hence, in the brain there are no 'global variables' or symbols. At the heart of the debate between connectionism and symbolism thus lies a conflict about the origin of meaning: whereas symbolism takes globally prespecified meanings of variables for granted, the objective of connectionism is (or should be) to explain the meanings of the primitive processing units from direct physical causes.

Accordingly, the term connectionism will be taken here to imply a functional *constraint* imposed on cognitive models, restricting what *interpretations* are permitted of the primitive units of the system. This constraint, which I will refer to as the *connectionist constraint*, can be stated as a necessary, but not sufficient condition for designing connectionist systems. It entails that *the interpretation of any of the system's primitive units* (as far as it is invoked in explaining experimental results) *must depend exclusively on processes that are handled internally and autonomously by the system, and must not appeal to any externally or globally imposed interpretation of the network units.* Hence, interpretations of the primitive units can be defined exclusively with respect to their topological position in the network (i.e., through inputs of efferent nodes), and eventually derived from (and grounded in) the external input. In section 4.3.1 I will argue that certain popular models, that are apparently able to deal with the problem of encoding the constituent structure of language, and that are presented as connectionist models, violate this constraint.

4.3 Neural networks with explicit representation of compositional structure

The ability to 'parse' the structure of a sentence in order to discern its phrasal composition, disambiguate the sentence and afford a semantic interpretation is an integral part of language processing, and it requires a processing model that is capable of compositional structure representation. The first connectionist reply to Fodor and Pylyshyn is to build compositional structure explicitly into the network. Representing compositional (or hierarchical) structure has however proven to be a hard, if not impossible task for connectionist networks. Often the task is phrased as a *variable* binding problem: how does the network solve the task of

binding fillers to roles, or tokens to types? For instance, one can represent a tree structure by assigning variables to each of the 'roles' in a tree, for example the root node, the left and right daughter of the root, the left daughter of the right daughter, etc. The tree is then represented by binding a symbol to each of the roles. Two variations of this approach to representing compositional structure are RAAM [Pollack, 1990] and Harmony Theory (HT) [Prince and Smolensky, 1997].

4.3.1 Recursive Auto-Associative Memory

The Recursive Auto-Associative Memory network (RAAM) was proposed by Pollack [1988] as a response to the critique of Fodor and Pylyshyn [1988] that connectionist models cannot operate with and represent compositional, recursive structures. The RAAM network is able to encode a tree structure of a sentence in compressed form in a single hidden layer, and later decode it to its original form, while recovering the constituents.

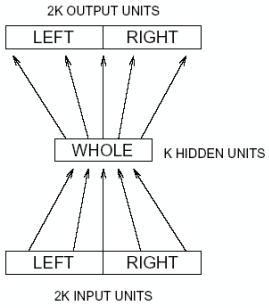


Figure 4.3: Recursive auto-associative network consisting of a compressor and a reconstructor component. Reproduced from [Pollack, 1988].

In the encoding phase, a binary tree is fed to the input nodes, level after level and from inside out, whereby at each step the trained hidden unit representation of one level of the tree is recursively fed back to the correct input (either the left or the right child). Training is done by teaching the auto-associator to reproduce the input activation on its output units, using error back-propagation. For example, suppose RAAM learns to encode the tree *(X John (Y likes Mary))*, then first *likes* and *Mary* are fed to the left (L) and right (R) input units respectively. Then, *John* is fed to L, while simultaneously the activation of the hidden layer from the previous step is copied to R.

As the network learns to reproduce the identical constituents on its output units (through *auto-association* and error back propagation), it forms a compressed internal representation of the combined left and right child in its k hidden units. During decoding the reverse process is followed: RAAM reconstructs the original tree step by step, starting with the complete compressed tree, and feeding

the output of the network back to its input, until the leaves of the tree have been reconstructed.

RAAM is not connectionist

Although RAAM is often cited as the textbook example of how connectionist models can solve the problem of encoding constituent structure, a critical analysis of the solution offered by RAAM reveals that it in fact violates the *connectionist constraint*, as formulated in section 4.2.1. The reason is that RAAM does not assign tree structure to sentences *autonomously*, but requires an external control structure, that maps the output of the compressor module to the left or the right input node during the encoding phase. The controller appeals to external knowledge of the syntactic structure in order to interpret the output as either the 'left child' or the 'right child' in the tree. In parsing terms this means that the 'stack', or memory of the system is taken care of externally. Since the system has no *local* access to the information needed to decompress the compressed tree, RAAM does not conform to the constraint of locality of interpretation, implying that the proposed solution could never be realized by an autonomous neural system.

4.3.2 Filler-role binding using the tensor product

Smolensky [1990] proposes an alternative solution for bridging the gap between representations of hierarchical tree structures at the symbolic level and at the connectionist level, within the Integrated Connectionist/Symbolic Cognitive Architecture (ICS) [e.g., Prince and Smolensky, 1997, Smolensky and Legendre, 2006]. He proposes to represent variable bindings between roles and fillers as tensor products, whose numeric entries can be mapped to units in a connectionist network. A node in a parse tree is encoded as the tensor product of a *role* vector, which represents the node's position in the hierarchy of the tree, and a *filler* vector, representing the node label. Let r_0 and r_1 denote the left child and the right child role in the tree respectively, then any tree position can be construed by recursive application of the tensor product. For instance, $r_0 \otimes r_1$ represents the right daughter of the left daughter of the root. An arbitrary syntactic tree can thus be encoded using vector addition of the tree nodes. For instance the tree (X A (Y B C)) would be represented as $X + A \otimes r_0 + [Y + B \otimes r_0 + C \otimes r_1] \otimes r_1$. The tensor product representation of a tree is subsequently mapped to a neural network by dedicating a unique network unit to every possible conjunction of a role and a filler (fewer nodes are needed if the role-filler encoding is distributed). Crucially, the mapping between syntactic trees and network units must be done manually.

The problem with this approach is of the same order as with RAAM: assigning network units to the tensor product representation of a tree appeals to knowledge

about the interpretation of network units, which is only available externally to the system. The ICS system cannot autonomously, or locally determine the correct mapping between tree nodes and network units; in practice, the interpretation of a network unit as a particular position in a parse tree (whether it is distributed or localist) is pre-given for any given network design. Since the extensional meaning of a unit is not arbitrary, the units function as *symbols*. Obviously, this violates the *connectionist constraint*, which basically expresses that connectionist systems should not make use of symbols.

A problem of a different kind for tensor product representations of bindings is that the number of bindings (hence, network units) that are required to represent an arbitrary tree node grows exponentially with the depth of the tree [e.g., Hummel and Biederman, 1992, Stewart and Eliasmith, 2009]. Moreover, the conjunctive bindings encoded in the tensor products cannot deal with novel sentences (with arbitrarily deep levels of recursive embedding) in a productive way, because every conjunction encodes a specific relation between roles and fillers in a single undivisable entity. As Hummel et al. [2004] point out, this makes conjunctive coding in general, and the tensor product in particular, unsuited for relational inference or generalization to similar events. As a consequence all bindings have to be specified in advance.

This problem relates back to the 'massiveness of the binding problem', mentioned in section 1.2: since language is productive, and given that the number of sentences that can be productively combined from an average person's vocabulary of several thousands of words is innumerable, there cannot exist enough specialized neurons in the brain to bind every sentence from its constituents.

In order to deal with novel sentences it is necessary that the language system can combine words or phrases in flexible ways. To achieve this, it has been proposed that the brain uses a mechanism for *dynamic* binding, rather than *static*, or *conjunctive* binding [Hummel and Holyoak, 1997, Hummel and Biederman, 1992, van der Velde and de Kamps, 2006] (see also sections 2.3 – 2.5). A dynamic binding solution is implemented by [van der Velde and de Kamps, 2006], whose work I will discuss next.

4.3.3 The neural blackboard architecture

In the neural blackboard architecture (NBB) van der Velde and de Kamps [2006] try to address Jackendoff's 4 challenges (see section 1.2), in particular the 'massiveness of the binding problem', and the 'problem of two'. In the NBB words are represented by word assemblies [Pulvermüller, 1999], which are grounded and never duplicated. Phrasal categories, such as NP and VP are represented by an unlimited reservoir of (syntactic) structure assemblies. Word assemblies can be bound to structure assemblies (through sub-assemblies such as agent and theme) to form sentences. A sophisticated electronic circuit involving memory gates and association gates regulates the direction of flow through the assemblies, such that

for example *cat chases mouse* can be distinguished from *mouse chases cat*. The massiveness of the binding problem is solved by using temporal (or dynamic) binding, as was also argued in section 2.5. The problem of two is circumvented because, if a word occurs more than once, it can be temporarily bound to multiple structure assemblies (NPs) from the reservoir, each of which tags the word a unique label.

The problem with the NBB is that only the word assemblies are grounded; the syntactic structure assemblies are not. Structure assemblies for NP's and VP's are simply assumed to be innately given together with a label, and the fact that they can bind with word assemblies of the same label is a direct consequence of the arbitrary external assignment of a label to the syntactic units. This violates the connectionist constraint, hence the NBB is not a viable connectionist solution. Further, although van der Velde and de Kamps [2006] rightly argue that grounded units cannot be duplicated, the NBB allows structure assemblies to be freely duplicated. This means that the problem of two is merely shifted to the phrasal level rather than being solved.

4.4 Recurrent, distributed networks

One class of connectionist models, the Recurrent Neural Networks (RNN, for short) is particularly suited for language processing because, unlike standard feed forward networks, these networks can compute functions for inputs of varying length, such as sentences, when they are presented word by word. As the name implies, recurrent neural networks have recurrent connections that copy activation among units from an earlier point in time to the current input. They therefore satisfy

$$x_j(t+1) = \mathcal{F}(net_j(x_i(t), w_{ji})) \tag{4.3}$$

where $x_i(t)$ is the current state and input at time t, $\mathcal{F}$ is an activation function, for instance the logistic function, and net_j the excitation function that integrates the input to x_j, for instance $net_j = s_j$, the weighted input given in Equation 4.1.

Several theoretical results about the *representational power* of recurrent networks demonstrate that they can in principle exceed the representational limitations of finite state automata, given unbounded precision of their weights and/or activation. Pollack [1987] showed that heterogeneous, second order RNN's with unbounded precision of the weights are Turing equivalent, which entails that they can represent any arbitrary function;[1] Siegelmann and Sontag [1991] showed that even finite, homogeneous first order RNN's with rational weights are Turing equivalent. This implies of course that a natural language such as English, if viewed as a characteristic function, can in theory be recognized by an RNN.

[1] An RNN is heterogeneous if not every unit has the same activation function; An RNN is of second order if the excitation function depends on products of inputs

In general, the strong representational power of recurrent networks with continuous activation values, which is a result of their ability to *represent* an unlimited number of states, does not guarantee that they are capable of *generalization* to novel sentences. There exists a risk of *overfitting* the model to the training data, leading to bad generalizations on test data. Yet, precisely the question about a model's capacity for generalization will ultimately be of interest to us when we discuss whether the family of recurrent networks is able to model the systematicity and productivity of human language (section 4.7.1).

4.4.1 The Simple Recurrent Network (SRN)

The most widely used recurrent network is Jeffrey Elman's simple recurrent network (SRN, for short) [Elman, 1989]. The SRN consists of a multi-layer feed

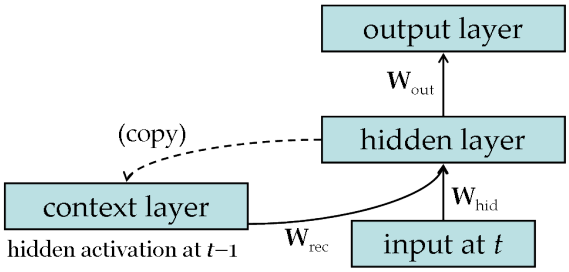


Figure 4.4: Elman's simple recurrent network (image design: Stefan Frank).

forward network with an extra so-called context layer (see Figure 4.4). In the standard setup, when doing a sentence processing task, every word from the training vocabulary has a dedicated input and output node, and sentences are presented to the input units word after word, separated by end of sentence symbols. The hidden layer units are connected through recurrent connections to the context layer units, and after every time step the activation of the hidden layer is copied back to the context layer. At the next time step the new input is combined with the activation of the context layer and both are forwarded simultaneously to the hidden layer. Through this feedback, the hidden layer learns about its previous internal representations, and can form a representation of the history of the sequence.

Apart from the weights between the hidden and the context layer, which are fixed at a value of 1, all other network weights are learned in a semi-supervised way, by presenting the input of time $t + 1$ as target to the output layer of the network at time t, and using error back-propagation (gradient descent learning).

4.4.2 Second connectionist reply to Fodor and Pylyshyn: distributed connectionism

As mentioned before, the second connectionist response to the structure encoding problem is that syntactic knowledge can be encoded 'implicitly' in the hidden unit representations of distributed recurrent networks. Note that thereby proponents of the distributed network approach dismiss the importance of what according to many linguists is one of the main functions of grammar: the structural analysis ('parse') of the sentence, and its disambiguation, facilitate a semantic interpretation. Parsing is considered a first step in the process of understanding the meaning of a sentence.

By their nature distributed networks are not well suited for explicit structure representation, because it is hard to represent structural relations between primitive categories if the latter are encoded in a fully distributed fashion. Yet, many experiments seem to show that the Simple Recurrent Network performs well on tasks for which knowledge of constituent structure is required, such as next word prediction across constituent boundaries, without taking recourse to explicit representations of syntactic categories, or constituents [e.g., Elman, 1990, 1991]. A concern is however that in this literature it is never formulated exactly and clearly what criteria a systematic model of language processing must satisfy. In the following sections I will make an attempt to formulate precise criteria against which the performance of connectionist models of language can be evaluated.

4.5 Defining criteria for the systematicity of language

4.5.1 Weak and strong systematicity according to Hadley [1994]

Recall Fodor and Pylyshyn's [1988] critique of connectionist networks, namely that they fail to account for the combinatorial productivity and systematicity of natural language. Systematicity and productivity entail an ability to comprehend or produce an unbounded number of novel sentences based on a limited number of observed sentences. For instance, if a speaker of English is able to comprehend one sentence (e.g., *John loves Mary*) then she is able to comprehend all other sentences which are structurally related (e.g., *Mary loves John*)

However, what exactly is meant by systematicity is never formalized by Fodor and Pylyshyn [1988] in a sufficiently precise way, such that it can be invoked as an operational definition to evaluate candidate connectionist models. It has proven difficult to give an operational definition of human linguistic systematicity that avoids any reference to global variables and formal operations, and that is devoid of any theory driven assumptions about the underlying representations.

To date, the most widely accepted operational definitions for evaluating the systematicity of connectionist models of language are those proposed by Hadley [1994]. His systematicity tests focus on the degree of generalization of various systems under learning conditions, in which the novelty of test sentences is varied relative to a training corpus. This captures the insight that the ability for *generalization*, rather than representation, is a key aspect of systematic behavior. Hadley [1994] distinguishes between the notions of weak systematicity and strong systematicity:

- A system is *weakly systematic* "if it can process sentences that have novel combinations of words, but these words are in the syntactic positions they also occurred in during training" [Hadley, 1994, p.7].

- A system is *strongly systematic* if, besides weak systematicity, it also "can correctly process a variety of novel simple and embedded sentences containing previously learned words in positions where they do not appear in the training corpus." [Hadley, 1994, p.7].

Weak systematicity does not account for the empirical fact that people are apparently able to process words in novel syntactic positions, as in *John loves Mary* versus *Mary loves John*. This means that knowledge of language must include some notion of sentence structure that is context independent. The latter aspect of systematicity is accounted for in *strong systematicity*.

Hadley reviewed the literature on several connectionist models, among which the SRN, RAAM [Pollack, 1988], and Smolensky's tensor products [Smolensky, 1990], and concluded that strong systematicity has never been tested on these models. Much ensuing research has focused on establishing whether SRNs are weakly or strongly systematic in the sense of Hadley [1994], with some authors claiming SRNs display strong systematicity [e.g., Christiansen and Chater, 1994, Brakel and Frank, 2009], and others disputing even weak combinatorial systematicity of the SRN [e.g., van der Velde et al., 2004].

Weaknesses of Hadley's definitions of systematicity

Unfortunately, there are several problems with Hadley's systematicity tests.

- First, underlying any practical definition of systematicity is a presupposition about the existence of *substitution classes* of linguistic expressions over which the systematicity is observed. Without the notion of a class of expressions that can be substituted for each other there can trivially not be generalization, hence no systematicity either.

  If strong systematicity evaluates whether a system generalizes across particular substitution classes (of words or other expressions), then its definition must explicitly acknowledge the a priori assumption of such substitution

classes and how they are used in the evaluation, otherwise systematic behavior is plainly unobservable. This may be trivial if systematicity is evaluated on an artificial language such as used in Hadley's definition, but if strong systematicity purports to have anything to do with the systematicity of natural languages, then explicit reference to substitution classes cannot be omitted from the definition.

- The second problem is that Hadley's strong systematicity applies exclusively to lexical categories, and ignores relationships between larger chunks of words. This is a too restricted interpretation of the human ability for systematicity, because it does not incorporate the fact that for instance anyone who understands *the brother of John loves Mary* also understands *Mary loves the brother of John.* A correct interpretation of the claims of Fodor and Pylyshyn [1988] concerning systematicity would have to focus on *constituents* rather than on words, since the empirical facts reflect an ability to substitute constituents of the same class in different sentence positions.

4.5.2 A proposal for a concise definition of the systematicity of language

To accommodate for the weaknesses of Hadley's systematicity criteria identified above, I propose a novel, and more concise set of criteria for evaluating the systematicity of connectionist models

1. For evaluating systematicity one must assume that
   (a) the participating units in a systematic relation are constituents[2] that can be larger than single words, and
   (b) there exist substitution classes of constituents over which the systematicity is observed, otherwise it cannot be evaluated (i.e., it is unfalsifiable).
2. Systematicity is defined as the property that given the constituents of (part of) a sentence their substitution class membership alone predicts the class membership of possible subsequent constituents. To the degree that natural language is systematic, it possesses sets of constituents (words or phrases) that behave as clean substitution classes.[3] I will refer to this as the *context invariance criterion.*

[2]The term constituent is intended here in the naive sense, and refers to any contiguous sequence of words ('chunk') in a sentence

[3]The extent to which a particular language possesses clean substitution classes is an open research question. In practice, of course, word categories and constituent classes are never perfectly 'clean', but class membership is graded, and so is substitutability (see section 3.3.1 for a discussion of the gradedness of syntactic categories).

3. Systematicity in natural language also entails that single words can be substituted for multi-word constituents of the same class. As a consequence constituents can be 'pumped up' (for instance, by recursively nesting relative clauses), while their class membership is preserved. I will refer to this as the *recursive systematicity criterion.*

4.5.3 The importance of inductive bias for generalization

The above analysis brings to the surface an even deeper problem with testing the ability of a system to generalize: before concluding that an unseen instance generalizes to a class of sample instances, one needs to have a notion of similarity. Yet, it is not possible to define similarity in absolute terms, since it depends on an underlying representation of structure that is inherent in the system. Similarity is by definition a relative notion.

To make this point more precise it is useful to reiterate an important observation about generalization from Mitchell [1980]. Consider a learning algorithm L that is trained on a finite number of training instances. It is a known fact that in order to be able to classify an unseen instance x_i it is necessary that the learning algorithm has a built-in bias that imposes structure upon the hypothesis space. There are infinitely many hypotheses that are consistent with a finite sample. For instance, the series $x_1 = 1,\ x_2 = 3,\ x_3 = 5$ is not only consistent with the hypothesis $x_{n+1} = x_n + 2$, but also with the hypothesis that $x_{n+1} = 7 \cdot x_n - 4^n$, which 'generalizes' to $x_4 = -29$.

This means that the kinds of generalizations that are permitted are underdetermined by the data alone, and are restricted only by the bias of the learning device. Without such a bias every hypothesis is equally likely. Quoting Mitchell [1980]: "Only if a system has ... biases for choosing one generalization over the other, can it non-arbitrarily [read: *systematically*] classify instances beyond those in the training set."

This applies to generalization in language as well. Suppose one has seen *John loves Mary* and *Mary loves Sue*, and identified these as sequences of the form *Noun Verb Noun*. Then, without any assumption about structure, there are many generalizations that are consistent with this example. For instance *John loves Mary loves Sue*, assuming that the rule is that nouns and verbs alternate. This means that there is no way to know from training data alone what is the 'correct' generalization. Rather, any judgment of generalization appeals to some *intuition* or assumption about how sentence structure is represented internally in the human language system.

Mitchell [1997] defines the *inductive bias* of a learning algorithm L as any minimal set of assertions B such that for any target concept c and corresponding training examples
$D_c = \{< x, c(x) >\}$ the classification of an unseen instance x_i follows by deductive

inference

$$\forall x_i \in X : B \ \wedge D_c \ \wedge x_i \Rightarrow L(x_i, D_c)$$

where X is the domain of L, and $L(x_i, D_c)$ is a classification of x_i after seeing the training data D. This means that knowledge of the inductive bias of a learning algorithm turns generalization into *deductive* (rather than *inductive*) inference from the training data.

The inductive bias of a system consists of a *structure bias*, which characterizes how the hypothesis space is structured. On top of the structure bias one can define additional background assumptions, for instance a *simplicity bias* that expresses a preference for simple models, using fewer rules. In algorithms for grammar induction often the principle of Minimum Description Length (MDL) is used as a simplicity bias (see sections 2.2.1 and 3.2.2).

There are two sides to a learning algorithm. The first is finding generalizations, based on similarity of training instances *with respect to the structure bias of a system*. The second is to consolidate the found generalizations in the model by *merging*, or *clustering* the training data into a more compact representation, driven by the simplicity bias. After a merge the internal representations of *similar* train instances become *identical*, resulting in an increased capacity for generalization of the system.

While the above discussion focused on discovering *similarities* between training instances, it seems that language (syntax) learning exploits the human capacity for discovering *analogies*. Analogy measures the similarity of relations between pairs (consisting for instance of two sentence analyses), and is therefore of a higher order than similarity between objects. A pair of relations is analogical, A:B::C:D, if the same or a similar transformation can be identified between A to B and between C to D, with respect to the structure bias of the system. By looking for analogical pairs in the training set a learner can discover an *induction step* (corresponding, for instance, to a rewrite rule), that is used more than once in the training set. In Chapter 8, particularly section 8.2.2, I will have much more to say about learning from analogy, in the context of episodic memory.

4.6 Systematicity and the Chomsky hierarchy

Mitchell's work on the relation between inductive bias and generalization is important for the systematicity debate, because it implies that by understanding the inductive bias of a language learning system, e.g., an automaton or neural network, one can infer what kind of generalizations one should expect. Specifically, applied to the Chomsky hierarchy, each family of grammars in the hierarchy predicts different generalizations.

A famous result in formal linguistics is that the grammar of a natural language such as English is richer than a finite state grammar, and within the Chomsky hierarchy occupies at least the place of context free grammars (see section 3.1.2).

Yet, many connectionists do not accept the presuppositions of Chomsky's proof about the unsuitability of the FSA as a model of English, because it is based on reasoning about *unbounded* recursion. Christiansen and Chater [1999] argue that the standard view of unbounded recursion is motivated by the assumption of a competence (phrase structure) grammar, rather than based on empirical observations of human performance. Models of language, rather than account for hypothetical behavior at infinity, need only explain finite, observable behavior. In their opinion the *explanandum* for a model of human-like recursion is people's limited recursive performance (so-called 'leaky recursion'), and their differential performance on different types of recursion.

The disagreement on the issue of unbounded recursion has resulted in the unfortunate circumstance that the Chomsky hierarchy has been largely ignored by connectionist researchers in the systematicity debate. In the next sections I will however make the case that, contrary to received wisdom among connectionists, the Chomsky hierarchy is still a very relevant reference point in the debate about the systematicity of language. I show that the objections of Christiansen and Chater [1999] can be circumvented by changing the point of view from the *generative power* of a family of grammars in the Chomsky hierarchy to the kinds of *generalizations* that it permits. If one considers the Chomsky hierarchy from the perspective of language learning and generalization there is no need to take recourse to assumptions about unbounded recursion. Mitchell's [1980] observation about the central role of the structure bias for systematic generalization links the systematicity debate to the well-understood field of formal grammars.

The structure bias of formal grammars in the Chomsky hierarchy is characterized by the variables and rewrite rules they permit (or equivalently, the inductive bias of an automaton by its possible states and state transitions). While the structure bias describes the generative power of the grammar, it also constrains what generalizations can be deductively inferred from a given training set. I will show in the next section that, for the same training set, as a result of their different structure biases, regular grammars and context free grammars capture different similarities between the train sentences, and therefore they make different generalizations. This justifies an approach which considers the observed systematicity of language as a means to find out something about the inductive bias of the human language system.

4.6.1 Evaluating the systematicity of formal grammars

Context free grammars are strongly systematic, but regular grammars are *not*

To illustrate the point of the previous section let us see what the effect of the structure bias of regular grammars and context free grammars is on their systematicity. Consider the following examples:

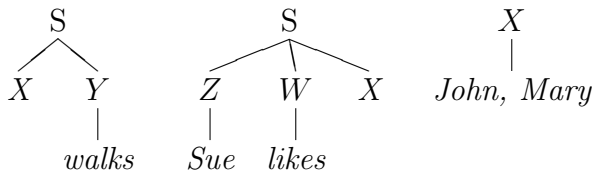


Figure 4.5: CFG grammar after learning from the first three examples.

(4.4) John walks.

(4.5) Mary walks.

(4.6) Sue likes John.

Our intuitions about *strong systematicity* [Hadley, 1994] tell us that these examples can be generalized to

(4.7) Sue likes Mary.

In a context free grammar (CFG) the analogy between the examples can be expressed by merging *John* and *Mary* into a single category X. This leads to the grammar of Figure 4.6.1, which produces the desired generalization ('Sue likes Mary').[4] Thus, the structure bias of a CFG conforms to the strong systematicity criterion.

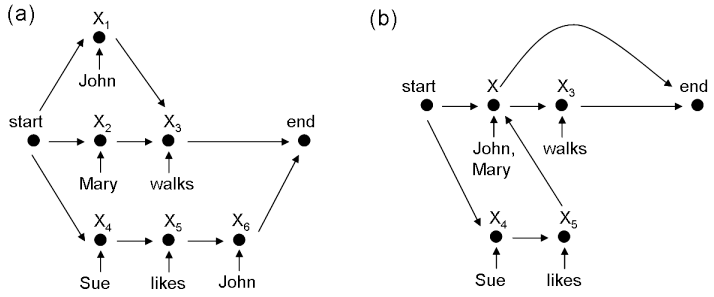


Figure 4.6: (a) Initial FSA after hearing three sentences. (b) FSA after merging X_1, X_2 and X_6 into X.

On the other hand, suppose the first three sentences from the example are represented using an FSA as the underlying automaton, as illustrated in Figure 4.6 (a), then merging the non-terminals that accept John and Mary will not only produce the desired generalization, but also a sentence as 'Sue likes John walks'

[4]Moreover, it can be shown that a learning algorithm such as BMM [Stolcke, 1994] (section 3.2.2), which has a CFG bias, will under reasonable initial conditions converge to the correct grammar, provided it is supplied with a simplicity bias.

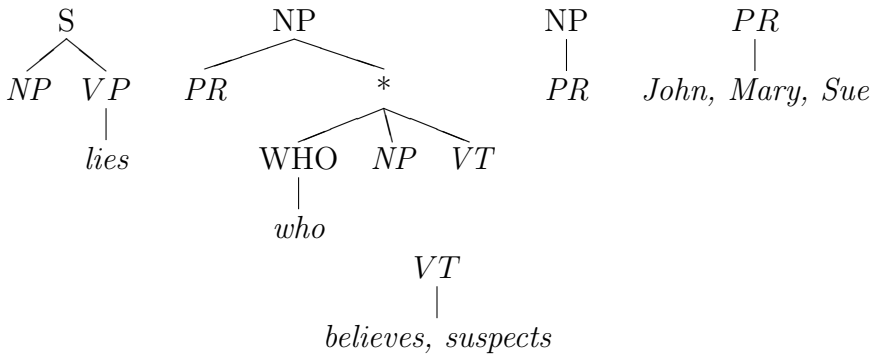


Figure 4.7: Recursive CFG generating *John who Mary who Sue hates likes walks.*

(see Figure 4.6 (b)). In fact there is no way for an FSA to produce exclusively the desired generalizations (as the FSA cannot distinguish between the John/Mary category in subject or object position), except for the trivial case where nodes are duplicated for every sentence. Yet, the latter grammar cannot be inferred deductively from the training data (i.e., given the structure bias). Hence, the FSA does not satisfy strong systematicity.

Context free grammars exhibit recursive systematicity, but regular grammars do *not*

The second case of interest concerns generalizations that are produced by nesting clauses:

(4.8) John lies.

(4.9) John who Mary believes lies.

(4.10) Mary lies.

(4.11) Mary who Sue suspects lies.

As was argued in section 4.5.1, a reasonable criterion for the systematicity of language is *recursive systematicity*, which predicts that English speakers can generalize the examples to

(4.12) John who Mary who Sue suspects believes lies.

Assuming the underlying representation is a CFG, then from analogy between the first and second pair of examples a similar transformation can be discovered that applies to both pairs, and expressed as a rewrite rule of the form $NP \rightarrow PR\ who\ NP\ VT$. This leads to the grammar of Figure 4.7, which indeed produces the correct generalizations.

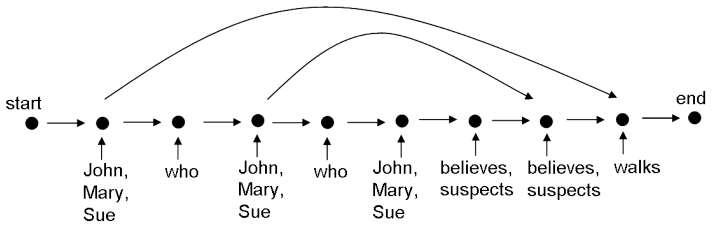


Figure 4.8: Trivial FSA that generates *John who Mary who Sue hates likes walks.*

An FSA could 'generalize' to the final example by copying nodes, as illustrated in Figure 4.8, repeating non-terminals for *who*, *John*|*Mary*|*Sue* and *believes*|*suspects* at every recursive level. However, this trivial solution does not follow deductively from considerations of analogy, and therefore it is not a true, non-arbitrary generalization in the sense of Mitchell [1980]. Hence, the trivial solution can only be found by coincidence.

Since duplicating nodes is not a fair strategy, the only alternative solution that generates all 5 sentences is the FSA given in Figure 4.9, but this FSA obviously overgeneralizes. In sum, the FSA does not satisfy the recursive systematicity criterion.

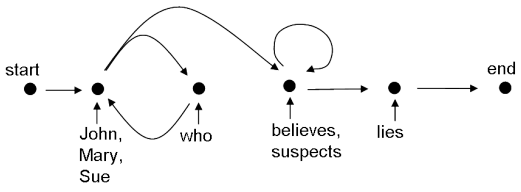


Figure 4.9: Overgeneralizing FSA for *John who Mary who Sue hates likes walks.*

4.6.2 A systematic model of language must have a structure bias that is at least as expressive as CFG

The previous sections show that the Chomsky hierarchy is in fact a theory about systematicity, that orders systems with different structure biases according to how well they generalize. This perspective allows formulating the properties of regular and context free grammars without making reference to unbounded recursion, hence it is not susceptible to the critique that has been mounted against the argument from the generative power of context free grammars [e.g., Christiansen and Chater, 1999]. The conclusion is that, contrary the accepted opinion, constraints from the systematicity of language (strong systematicity and recursive systematicity) provide strong evidence that the human language system has as least a CFG bias. A more general conclusion is that, in my opinion, it does not make much sense to distinguish between the notions of systematicity and syntax.

Both refer to the same phenomenon, namely that part of linguistic behavior is predictable from structural regularities alone.

There are additional and independent arguments for the need for hierarchical representations, based on general considerations of learning strategy. Models of language processing that have the same structure bias as the FSA (for instance the SRN: see the next section) make an implicit assumption that humans have only a general associative learning mechanism, in which learning is driven by the discovery of transitions between words [e.g., Saffran et al., 1996]. However, a more useful approach from a cognitive perspective conceives of learning as a process of discovering differences between stimuli, that is, learning is driven by discrimination [e.g., Steels, 1996]. Quoting Edelman [2008, p. 250] "the only computationally viable approach [to learning a language from a continuous sound stream] is to seek a partial match between two entire signals, and to decide that each of the non-matching parts is a candidate for an independent symbol". In other words, since only the distinguishing features in the speech signal give information about the relevant variation in language, it is most efficient to store only those.

If the brain uses such a discriminative approach to learning it follows that it employs hierarchical representations. Discriminative learning requires at least a context free bias, because in order to discover a non-matching part between two strings, $A\ M_1\ B$ and $A\ M_2\ B$, one must compare the strings and detect common contexts A and B. To do so, the system must be able to align the strings and hypothesize rules of the form $X \rightarrow A\ \ M\ \ B$, where M can be any variable length string. Context free grammars do exactly this, hence they are well-designed for a learning strategy that is based on the discovery of differences. On the other hand, regular grammars are badly designed for the task of discriminative learning.

4.7 Systematicity and the Simple Recurrent Network

4.7.1 The structure bias of the SRN compares to that of an FSA

In order to identify the structure bias of neural networks one must focus on their *states* and *state transitions*. In an automaton a state is a theoretical construct that captures the internal knowledge of a device about relations between inputs and outputs. In a recurrent neural networks, such as the SRN, the state of the network at time t denotes the simultaneous activation values of the hidden layer units.

A closer look at the state transitions of the SRN reveals that the hypothesis space of the SRN shares its structure bias with the family of regular grammars.

The activation functions that determine the state transitions in an SRN are of the same form as the rewrite rules of an FSA: whereas rewrite rules of an FSA are of the form $A \rightarrow w$, or $A \rightarrow w\ B$ (where w is a terminal symbol and A is a non-terminal symbol), the activation functions of the SRN are functions of the input plus at most one earlier (hidden) state. Hence, given the same training data, one expects the SRN to be able to make the same *induction steps* as an FSA (hence to learn the same thing, despite the fact that its states are continuous).

A further argument for the kinship between the SRN and FSA is given by Jacobsson [2005], who proves that discretization of the state space of an SRN converts it to an FSA. Servan-Schreiber et al. [1991] show that the SRN, which they refer to as a 'Graded State Machine', can be trained to closely mimic a simple FSA (the Reber grammar), such that clusters of the hidden layer states correspond to states of the FSA.

The fact that the SRN has the same structure bias as the FSA implies that

- The SRN is not strongly systematic
- The SRN cannot capture recursive generalizations over constituents

One obvious difference between learning in the SRN and in an FSA is that whereas discrete state automata produce generalizations by means of a merge operation, distributed recurrent networks, having a continuous state space instead, produce generalizations by *clustering* states in state space. However the continuity of the state space should not give the SRN any advantage over the FSA in its learning strategy, nor is it relevant what particular learning algorithm is chosen (e.g., gradient descent), because as was argued in section 4.5.3 eventually the only factor that influences how a system generalizes — in a non-arbitrary way — is its inductive bias.

Yet, in several publications it is claimed that results obtained with the SRN on a word prediction task indicate strong and/or recursive systematicity [e.g., Elman, 1990, 1993, Rodriguez et al., 1999, Christiansen and Chater, 1999]. I will investigate these claims in Appendix A.

There exists a long list of connectionist models of language that either use the SRN as a component in a hybrid system, or share the inductive bias of the SRN (i.e., exhibit Markovian behavior). To name some, without being exhaustive: Long Short Term Memory [Hochreiter and Schmidhuber, 1997], SARDSRN [Mayberry III and Miikkulainen, 1999], SRSOM [Voegtlin, 2002], RECSOM [Koskela et al., 1998], STORM [McQueen et al., 2005], Fractal Encoding Networks [Tabor, 2000]. I will not discuss any of these models here, but note that with respect to the recursive systematicity criterion they fall in the same category as the SRN.

4.7.2 The SRN does not satisfy the context invariance criterion

Recall that the *context invariance criterion* of systematicity states that a systematic model of natural language should make predictions based on constituent class membership alone, and independently of literal preceding context. Because of its built-in sensitivity to context the SRN cannot make predictions based on the class membership of an arbitrary substring. Since the SRN cannot cut off history at some point (i.e., it does not make context independence assumptions as formal systems do), the contextual input always goes back to the first word of the sentence (or further). Yet, to make a correct next word prediction, the human language system often must base its decision on how the sentence is broken up into constituents. Consider the following examples

(4.13) I [told [her parents] ...

(4.14) I [[told her] [parents ...

Both prefixes produce exactly the same state in an SRN, hence the SRN cannot differentiate between them. Yet, prediction of the sentence continuation critically depends on the relations between the hypothesized constituent classes, e.g.

(4.15) ...that she is a very smart girl.

(4.16) ...are annoying people.

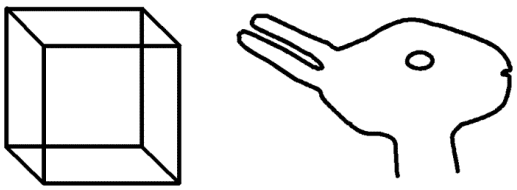


Figure 4.10: Ambiguous figures. The Necker cube and the duck-rabbit.

Whereas in the SRN the two interpretations of the prefix share a single representation or state (the so-called principle of *superposition*), there is evidence that our cognitive system is in a different state for each of the interpretations. This is related to the mind's ability to perform structural disambiguation, which is lacking in distributed neural networks. In visual processing structural disambiguation is known under the 'Gestalt laws of perception'. These formulate the idea that in perception, like in the examples above, 'the whole is not the sum of the parts'. The ambiguous figures in Figure 4.10 illustrate that people jump between different interpretations when looking at a single stimulus. Several experiments have found neuro-physiological evidence for discrete jumps between concepts for

a single percept, for instance in binocular rivalry in monkeys [e.g., Leopold and Logothesis, 1996].

An alternative formulation of the context invariance criterion of systematicity is that a systematic language device must be able to encode and work with relationships between abstract categories, what Hadley [1994] calls *strong* systematicity. In the absence of strong systematicity a language model is forced to memorize all lexical relations separately, instead of summarizing them in abstract relations. van der Velde and de Kamps [2006] note that the SRN, because of its perceived lack of (even weakly systematic) combinatorial productivity has to deal with a combinatorial explosion of word sequences, and therefore the SRN can only work with artificial languages with a very limited vocabulary (about 20 words), and not with natural languages.

Note that the difficulty with encoding abstract knowledge is not a specific problem of the SRN, but applies to distributed networks in general. The fact that distributed networks cannot encapsulate information within invariant units precludes them from encoding abstractions and storing knowledge systematically.

4.7.3 Locality of learning algorithms prevents true generalization of distributed patterns

The lack of 'context invariance' of distributed networks is also the focus of the critique in Marcus [2001]. Marcus showed that if categories are represented in a distributed manner, i.e., encoded over multiple feature nodes, then they do not behave like 'variables', in the sense that they do not generalize the input-output relation to all members in the domain of the category. This assumes that the category can take on any value that can be encoded on the input nodes (it is 'universally quantifiable'), and that the similarity between representations over multiple input nodes is preserved during mapping from input to output.

Although Marcus' argument is quite elaborate, the bottom line is this: because of the behavior of the learning algorithms in most networks (in particular error back-propagation), variables that are encoded over multiple nodes do not show the behavior that should be expected from variables: the nodes do not operate as a unity (they do not behave as a single variable), but each node learns independently from the other nodes. This has to do with the fact that learning is local: the weight adjustments depend only on the activation of the local neighbor nodes. For example, if one of the set of input nodes that together encode the variable is never activated (because of an arbitrary choice of training samples), then the connections emanating from it are never changed, irrespective of what happens to the connections from the other nodes that represent the same variable. In this case, the network has no way to tell that all nodes representing the variable should be treated uniformly. The conclusion is that a network that employs a distributed representation of variables cannot generalize the learned input-output

relations across nodes. Marcus' argument of course does not apply to localist networks, which can represent variables (prototypical categories) using a single node. Surprisingly, Marcus [2001] easily dismisses this possibility, yet the model presented in this thesis will use localist representations as a starting point.

Although I agree with Marcus on the premiss that distributed networks cannot learn certain rules, I disagree with his conclusion that the language system must therefore have a dual nature — consisting of a connectionist module and a symbolic rule-based system [e.g., Marcus et al., 1999]. In this thesis I will advocate an approach to modeling cognition that reconciles structure sensitive (but not symbolic) rules with a statistical way to learning not only the probabilities, but even the extensional meanings of the rules. As the HPN model, described in the next chapter, will show, certain rule-like primitive structures that contain placeholders for variables (slots) can be learned from distributional information alone — conform the connectionist constraint (i.e., with no innate assumptions about meaning, but only an innate architecture).

4.7.4 The systematicity of the SRN in the literature

In the previous sections I have argued on theoretical grounds that the SRN is not suited as a model for systematic language processing, because it does not satisfy either the context invariance criterion or the recursive systematicity criterion. Nevertheless, claims to the contrary are abound in the literature. For instance [e.g., Elman, 1990] alludes at strong systematicity by claiming that the SRN encodes knowledge of abstract word categories implicitly. Further, in [Elman, 1991] it is claimed that the SRN can *systematically* represent grammatical knowledge about constituent structure implicitly, using the displacement of the hidden states in state space to encode depth of embedding. This seems to suggest that the SRN satisfies the criterion of *recursive systematicity*.

In Appendix A I will critically scrutinize those claims in two case studies. The gist of my critique is that neither the so-called 'implicit' word categories nor the knowledge of 'displacement' are internally encoded in a form that the SRN can access it. Hence, such representations are not causative of any behavior of the SRN, but rather attributed as post-hoc explanations for its behavior.

4.8 What the systematicity of language tells us about the connectivity of the brain

To summarize this chapter so far, from considerations of the systematicity of language two major criteria were identified that realistic connectionist models of language processing must satisfy (section 4.5.2): recursive systematicity, implying that the inductive bias of the system must be at least as expressive as a CFG, and

context invariance. Subsequently, it was shown that the SRN fails both criteria, and distributed networks in general fail the second criterion [e.g., Marcus, 2001].

One may wonder where this leaves connectionism as a cognitive theory? I believe the connectionist philosophy can still play a significant role in the linguistic debate, if one takes advantage of the lessons learned from the systematicity of language to design a better type of connectionist model. Language is after all perhaps our richest source of information about the structural organization of the brain.

In the remainder of this section I will argue that the systematicity debate forces us to rethink the nature of connectivity in the brain and consequently in neural networks. I will show that if one abandons the notion of activation spreading as the exclusive means of communication between units in a connectionist network, and if one accepts that network units can represent assemblies larger than single neurons, then there exist connectionist solutions both for the context invariance criterion and the recursive systematicity criterion.

4.8.1 The context invariance criterion

In section 4.5.2 I formulated the *context invariance criterion* of systematicity, which states that a systematic model of natural language should make predictions based on constituent class membership alone, and independently of literal preceding context. Formal, symbolic grammars satisfy this criterion by virtue of making independence assumptions. Independence assumptions allow to encode the knowledge of a grammar at an abstract level, as a set of rules over variables.

From the perspective of the automaton, context invariance is achieved by functionally separating the output of a state in the automaton from the literal content of its input — I will refer to this as *encapsulation* of states.[5] Encapsulation does for states what independence assumptions do for variables: it creates abstraction, or *invariance* by cutting off literal context at some point in history, and in doing so it factors out smaller, and more abstract units of production than entire sentences.

Encapsulation in connectionist networks

The key to bridging the gap between symbolic and connectionist systems, it seems, is to endow the latter with a capacity for *encapsulation* of their states.

For an idea of how this can be achieved one may draw inspiration from the brain's solution for creating invariant representations. As was explained in section 2.1.3, the mechanism that the cortex employs for this purpose also amounts to *encapsulation* of information, as it is relayed from lower levels to higher levels in the cortical hierarchy: according to the MPF [Hawkins and Blakeslee, 2004]

[5] In the systematicity debate the term encapsulation is often used to refer to the separation between the syntactic and lexical-semantic modules. This is not what is meant here.

columns in higher cortical levels create invariant representations of (sequences of) columns in lower level regions by representing the lower level sequence as a constant 'name'. The name then serves as a building block for larger and even more invariant patterns. Essentially, this means that the brain works with pointers.

An important claim of [Hawkins and Blakeslee, 2004] and of this thesis is that *encapsulation serves as the neural basis for abstraction.* This points at a fundamental conceptual difference with the distributed network approach. In distributed connectionism generalizations are based on the similarity of distributed representations, and implicit abstract 'categories' can be inferred from clusters in the hidden unit space. The distributed approach however cannot deal with higher order abstraction (which was argued to be required for systematic hierarchical language processing), because it is impossible to use clusters as building blocks in further generalizations.

Following [Hawkins and Blakeslee, 2004], I propose that encapsulation can be implemented in a connectionist network by letting the communication between units depend not (only) on their activation, which is by definition context sensitive, but also on some intrinsic (possibly acquired) representation of the unit, which is context invariant. In the latter case the predictive behavior of a unit depends on its identity alone, as required by the context invariance criterion.

To enforce encapsulation in a connectionist network one must thus introduce a functional separation between the input and the output activation of a unit, or in other words create a so-called abstraction interface.In the HPN model, discussed in the next chapter, this is achieved by having activated units produce a constant output pattern (learned through experience), that represents the unit's relative topological position in the network.

By means of encapsulation of their states connectionist networks can emulate the context invariance property of variables, without actually using variables, but by creating *invariants.* For the connectionist-symbolist distinction it is important to emphasize the difference between invariants and variables: while both function as 'placeholders', the extensional scope of a variable is globally specified in advance, or innately, whereas the extensional scope of an invariant is incrementally constructed in the course of learning through interaction with the external world.

Encapsulated units act as placeholders, because the same unit can bind to a range of different 'filler units' at its input. At the same time they are invariants, because their intrinsic representation (the 'name'), and hence their extensional scope, is learnable from (statistical) experience. (The next chapter illustrates these principles with the HPN model: units in HPN have 'slots', where they can bind to other units (implementing the placeholder function), and at the same time they have adjustable (hence, not innate) representations, which determine variable relations with other network units.)

This means that a connectionist, statistical learning mechanism *can* learn algebra-like rules that represent relationships between placeholders, arguing against

the suggestion of [Marcus et al., 1999] that a specialized, symbolic rule learning device must be assumed in addition to a statistical learning mechanism. It also means that, contrary to received wisdom [e.g., Fodor and Pylyshyn, 1988], connectionism and a combinatorial, rule-like behavior are not at odds.

4.8.2 Dynamic binding in syntax

For a complete solution for syntactic processing in connectionist networks it is still needed to provide an explanation of how encapsulated syntactic units connect, or *bind* to one another. In language, the *binding problem* is concerned with the question of how independent syntactic and lexical elements combine into a single, coherent structure representing a (parse of a) sentence [e.g., Jackendoff, 2002, p. 59]. The challenge arises from the combinatorial productivity of language, that is the ability to produce and comprehend an unlimited number of novel sentences. As was argued in section 4.3.2, fixed, conjunctive bindings, as used in activation spreading networks, cannot account for productive language use. Moreover, fixed connections are inherently activation-dependent, thus cannot support context-invariant operations between encapsuled units.

In order to support flexible, context-invariant syntactic operations it has been suggested that some form of *dynamic* binding of the units must be in place [e.g., van der Velde and de Kamps, 2006, Hummel and Holyoak, 1997] (see also section 2.4). The general idea behind dynamic binding is that units (e.g., neural assemblies) can be flexibly grouped in novel configurations, without a need for pre-existing, dedicated binding neurons (so-called 'grandmother cells'). To achieve this, most proposals for dynamic binding make use of 'labels', or 'tags', to group neurons, for instance the 'oscillation phase' in synchronous binding, or 'enhanced activity' (attention) in serial binding.

In section 2.3 I proposed that syntactic binding, or *substitution*, can be construed by analogy to a particular solution for dynamic binding, which has been proposed by Roelfsema [2006] as the brain's solution for visual contour binding. Recall that according to this proposal contours are serially bound in the visual cortex by the spreading of a label (of enhanced activation) through lateral connections between topologically neighboring neurons (see Figure 2.5 in section 2.3).

There is however a crucial difference between the visual categories involved in contour binding and lexical and syntactic categories. Whereas the former represent local, *perceptual* features (e.g., orientation or shape), the latter represent goal-driven *conceptual* features, which are defined by their occurrence in certain sentence contexts. In section 2.6.2 I argued that lexical categories reside in the 'conceptual poles' of cortical columns, while the corresponding acoustic category (i.e., the sound of the word) is located in the 'perceptual pole' of the same column. Columns thus simultaneously participate in a 'bottom-up' (perceptual) network and a 'top-down' (conceptual) network.

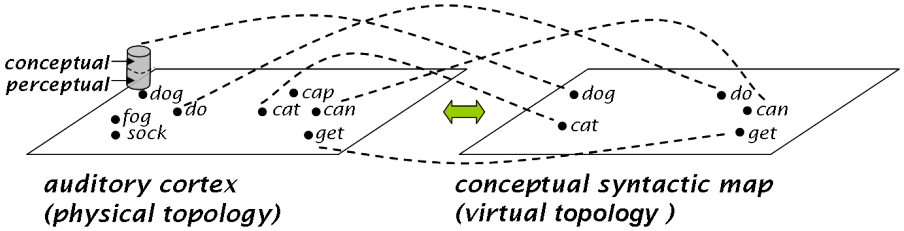


Figure 4.11: Columns representing words exist simultaneously in two topologies; one pole sits in a perceptual topology, and the other pole in a conceptual, or lexical topology.

For syntactic binding it would be desirable that the 'top-down' syntactic network also realizes a topology, since whether encapsulated units can dynamically bind should depend on their intrinsic vector representation alone. However, because the position of the columns is already fixed by their arrangement within an 'acoustic' topology (according to their perceptual poles; see Figure 4.11), this means that in general columns representing similar syntactic or lexical categories on their conceptual poles (e.g., *dog* and *cat*) are not also physical neighbors in the cortex. Therefore, the exact same solution for serial binding of visual contours, which makes use of lateral connections between neighbors in a perceptual topology, would not be practical for dynamically binding syntactic assemblies.

The question is thus whether a comparable solution as for serial contour binding can be found that operates within the 'top-down' syntactic network, and that dynamically binds syntactic assemblies with similar representations. The proposed solution must at least show that dynamic binding in syntax is not inconsistent with the connectionist constraint of locality, hence can in principle be implemented in neural hardware.

The switchboard construction

As a mechanistic explanation for syntactic binding I speculate that the brain makes use of a *switchboard* construction. This means that rather than being directly connected, syntactic assemblies are interfaced through a central hub that implements a kind of *telephone switchboard*. The function of the switchboard is to redirect a label (for instance, enhanced activation) from one assembly to another, of which the (topological) address is given by the sending assembly. By virtue of this central addressor system neural assemblies from diverse and physically remote areas in the cortex can flexibly connect.

An interesting possibility is that the switchboard interprets the 'names' (i.e., the intrinsic syntactic representations encoded in the name fields of the columns) as topological addresses. In that case the switchboard de facto realizes a *virtual* syntactic topology, based on the conceptual representations (addresses) of columns that are connected to the hub.

Although the idea of a switchboard may at first sight seem unlikely, it achieves many important goals at the same time, that could never have been achieved if neural assemblies were connected directly via hardwired lateral connections.

- *Sparse connectivity.* Indirect connectivity of the units via a central hub requires far less connections than a network design where all units are fully interconnected ($O(n)$ and $O(n^2)$ respectively, where n is the number of units). The hypothesis that the connectivity of the brain conforms to a so-called 'small world network' (which entails that connectivity is organized around several central clusters, such that any given cortical area can be linked to any other area with only a few projections) is supported by several anatomical studies and graph theoretic analyses [e.g., Young, 1993, Stephan et al., 2000].

- *Productivity.* The presence of a switchboard intermediating between assemblies provides the brain with an ability to flexibly bind previously unseen or unconnected neural assemblies. New assemblies can be easily integrated in the network without a need for full-scope rewiring (a known problem for distributed networks), and existing assemblies can be combined in productive ways.

- *Abstraction.* The switchboard provides an *interface* that separates the activitation of connected assemblies (i.e., it enforces *encapsulation*). This is a necessary condition supporting the ability to perform context invariant operations in the network.

- *Topological organization.* Because the switchboard connects assemblies based on their *addresses*, concepts can be represented with respect to a position in a 'virtual' topology, independently of individual neurons. Moreover, conceptual (and grammatical) knowledge can be expressed as relations between regions in a topology, making it much more robust against local damage or noise (see in this respect also the discussion in section 2.6.1).

- *Learning consists of topological self-organization.* Given a mechanism for updating the intrinsic representations (addresses) of columns connected to the switchboard, then learning becomes a matter of topological self-organization. It is far more efficient to update topological representations than to rewire the network.

- *Explicit encoding of binding history/network activity.* The bindings that are established in the network are made explicit as they pass through the switchboard. Since all binding operations are handled in one central place in a serial manner, the brain can record temporal activation paths through the network. This will be important for encoding episodic memories, which consist of sequences of successively activated assemblies. (The presumed

role of the switchboard in memory consolidation will be discussed in Chapter 8.)

In section 5.7.1 I propose a simple implementation of a switchboard in a connectionist network, that combines all the above features, while complying with the constraint of locality of interaction.

4.8.3 The recursive systematicity criterion

Connectionist models of language must also satisfy *recursive systematicity*, i.e., have at least the inductive bias of a CFG (section 4.5.2). In section 4.3 I discussed connectionist models that do have an explicitly built-in ability for recursive processing, namely RAAM [Pollack, 1988] and and the ICS architecture of Prince and Smolensky [1997]. However, both RAAM and the ICS crucially rely for their solution of encoding hierarchical structure on the assumption of global, and innate meanings of the network units (i.e., 'roles' that define the position in the tree), and this makes their approach intrinsically symbolic.

From philosophical considerations, I believe that the only way in which structure representation can be accomplished by an autonomous system such as the brain (i.e., in a way that complies with the *connectionist constraint* of section 4.2.1, and avoids reliance on external interpretations) is to use the internal hierarchical structure of the brain to map the hierarchical representations of sentences onto. In this case the structural representation of the system does not depend on any (external) interpretation, but can be read off from the internal configuration. According to this solution there is a stock of (initially blank) complex primitive units with an internal structure, resembling little 'treelets', that can be dynamically bound together into larger trees. Like the treelets proposed by Marcus [2001, p. 108], the complex units have registers, or local memories, that keep track of the internal state of the unit.[6]

The hypothesis of primitives (cortical columns) in the brain with internal hierarchical structure (emulating a CFG bias) is backed up by evidence from the functional architecture of the cortex. Hawkins and Blakeslee [2004] propose that cortical columns of higher regions represent temporal sequences of patterns from lower level regions (i.e., through hierarchical temporal compression). When they unfold such sequences implement a branching structure, similar to treelets.

In sum, on the basis of the review in this chapter it can be concluded that there is need for a radically new approach to connectionism that is more compatible with the hierarchical, columnar and topological organization of the cortex, with

[6]Marcus [2001, p. 108] makes the case for encoding hierarchically structured knowledge in the brain using a set of prestructured templates called treelets. The mind has a large repository of empty treelets on hand, that can be filled with new knowledge. Marcus' treelets possess register sets (local memories) where information, such as lexical meanings, can be stored. However, Marcus' treelets cannot be dynamically bound, and cannot learn, and as far as I know his ideas have not been implemented.

its treatment of invariant representations, and with the systematicity demands of language. I propose a new framework for connectionism called the Hierarchical Prediction Network (HPN), which will be presented in the next chapter.

Chapter 5

The hierarchical prediction network

The current chapter describes an instantiation of the neural theory of grammar that was proposed in Chapter 2, called the hierarchical prediction network (HPN). The HPN model demonstrates how the principles of dynamic binding *and* encapsulation *of states can be implemented in a connectionist network, allowing for productive, combinatorial operations. Phrase structure is accounted for in HPN by introducing complex network units that perform temporal compression of their inputs, incorporating the idea of hierarchical temporal compression from the Memory Prediction Framework. The presence of local registers in the network units enables HPN to function as a full syntactic (left corner) parser.*

HPN offers a connectionist account of the origin of syntactic categories, that does not depend on an innate specification of category labels. HPN assumes a continuous category space which dynamically adjusts itself to the linguistic input. Learning a grammar in HPN amounts to the formation of a syntactic topology through self-organization, based on neurally plausible principles. I show that as a result of learning, regions gradually emerge in the topology that reflect the traditional discrete syntactic categories. HPN is empirically evaluated on the task of semi-supervised grammar induction from bracketed sentences, both on artificial language and on realistic, spoken child language. Finally, I will discuss some limitations of HPN that motivate an extension of the model with episodic memory.

5.1 HPN architecture

To introduce the HPN framework, before presenting the formal model, it is helpful to approach it from traditional linguistic theory. Figure 5.1 illustrates some of the core design features of HPN and how they map to classical linguistic concepts. As shown in Figure 5.1 (a), the role of rewrite rules is played by complex units, so-called *compressor nodes*, that perform temporal compression on their inputs. The root of a compressor node compares to the left hand side of a context free rewrite rule, while its slots (the 'legs' in the Figure) correspond to the right hand side of a rewrite rule. An important difference is however that the slots of a compressor node are not associated with any particular non-terminal, and roots and slots have no meaningful labels.

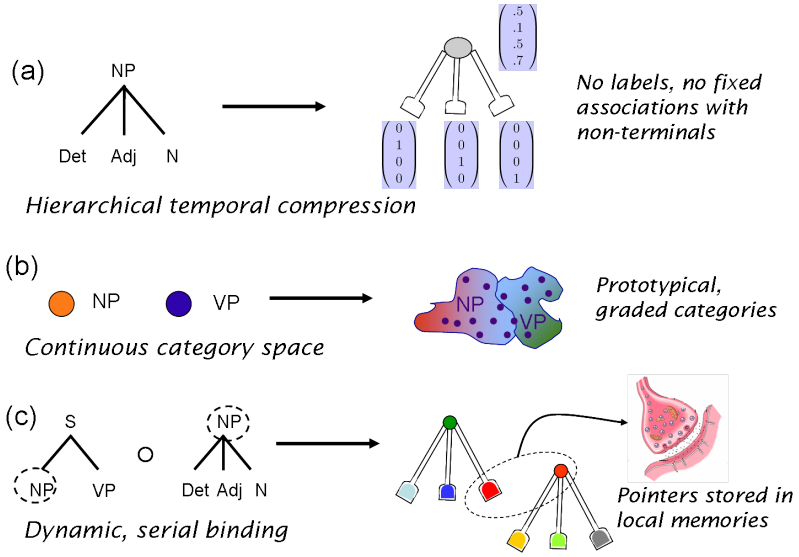


Figure 5.1: Core design features of HPN. (a) Hierarchical temporal compression is performed by complex units that temporally integrate a sequence of inputs, and which take on the role of traditional phrase structure rules. (b) Discrete syntactic categories are replaced by a large number of units that are situated in a continuous high-dimensional vector space. (c) Dynamic, serial binding functions as the neural correlate of substitution.

Figure 5.1 (b) shows that instead of employing discrete syntactic categories, the HPN network is initialized with a large number of units within in a continuous high-dimensional vector space. As the space self-organizes, regions in the space can be identified that correspond to conventional linguistic category labels. In Figure 5.1 (c) the symbolic operation of label substitution is shown to have a neural counterpart in dynamic, serial binding. Instead of a discrete notion of substitution mediated by label identity, in HPN substitutability between units is based on the proximity of their vector representations within the category space.

Unlike most other connectionist models, HPN assumes that the basic organizational and computational units of language processing are cortical columns rather than neurons [Mountcastle, 1997]. Consequently, one may expect that units in the HPN network possess some sophisticated properties that are not present in the simple McCulloch-Pitts neuron. Below the main components and terminology used in the HPN framework are listed. In section 5.7 I will present the neural motivation behind each of the components of the model.

Input nodes *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 observed within a sentence. Sentences are presented to the network one word at the time, and HPN processes them incrementally from left to right.

Compressor nodes Apart from input nodes, HPN also assumes so-called *compressor nodes*. Compressor nodes have a *root* and two or more ordered *slots*, with whom the root forms a fixed unit (see Figure 5.2 (b)). They owe their name to the fact that each compressor node dynamically binds a temporally compressed sequence of two or more nodes to its slots.[1] Hence, they encode sequences of words or phrases. A compressor node fires when all of its slots are bottom-up activated (bound) in the correct order.

Slots The *slots* of the compressor nodes function as sites in the network where the nodes dynamically bind to each other. Binding, or *substitution* is operationalized by temporarily storing a set of identical *tags* in the slot and in the root of an input or compressor node that is bound to it (see section 5.7 for the neural implementation).

Substitution space The *substitution space* is a high-dimensional vector space $\mathbb{R}^N$ that encodes substitutability relations between units. The roots of (input and compressor) nodes have vector representations with respect to this basis (see Figure 5.2). These representations are initially random, but after training the network they will come to reflect the distribution of the training data (see section 5.4). If learning has been successful, and the substitution space has self-organized, within its topology regions will emerge that correspond to conventional syntactic categories, such as nouns and verbs (see Figure 5.2). Thus, regions in substitution space define a continuum of categories, and a node's location in substitution space determines its graded membership to multiple categories. The slots of the compressor nodes represent independent substitution sites that span an orthogonal basis for the substitution space, hence its dimension N equals the total number of slots.

[1] Recall, that hierarchical temporal compression is a key element in the Memory Prediction Framework. Compressor nodes are an instantiation of this idea.

Dynamic binding and substitution Whereas in symbolic parsers rewrite rules combine through the formal operation of label *substitution*, in HPN units bind through serial, dynamic binding. This is realized by passing a *tag* between a bound node and the slot it binds to, involving a local, *material* operation and a switchboard construction (see section 5.7). Two identical tags that are stored in the slot and the node on two sides of a binding effectively implement a pointer, the so-called *path connector*. Thus, the *dynamic binding operator* connects activated compressor nodes and input nodes into a single connected path when a sentence is processed, implementing a neural correlate of substitution (see also section 5.3).

Local neuronal memories HPN assumes that slots have local memories, where tags are temporarily stored for as long as the parse process lasts.

Substitutability The probability of substituting a node in a slot, is given as a topological distance in substitution space. In the current implementation of HPN the inner product between the slot and the root representations is used as a measure of distance.

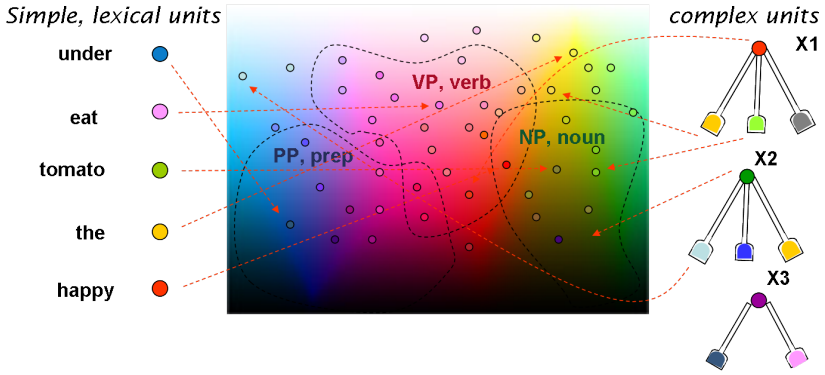


Figure 5.2: Schematic illustration of the substitution space. The overlapping regions corresponding to classical syntactic categories are imaginary. Input nodes are on the left, compressor nodes on the right.

The description of the HPN network can now be stated more formally:

Definition 1 (The hierarchical prediction network)**.** The hierarchical prediction network (HPN) is a 6-tuple $\langle V_T, L, C, \mathbb{M}, \circ, P\rangle$ where V_T is an alphabet of symbols, or words, called the *lexicon*; L is a set of *input nodes*, which are pairs of words $w \in V_T$ and *input node roots* $r \in R_L$ in $V_T \times \mathbb{R}^N$; (Here $R = R_L \cup R_C$ is defined as the set of roots of input nodes and compressor nodes.) C is a set of *compressor nodes*, which consist of a *root* $r \in R_C$ in $\mathbb{R}^N$ and two or more *slots* $s \in \mathbb{R}^N$ in S (the set of slots); $\mathbb{M}$ is a metric on $\mathbb{R}^N$, which is called the *substitution space*, on which a distance function is defined (the inner product $\langle \mathbf{x}, \mathbf{y}\rangle = \sum_i x_i \cdot y_i$); $\circ$ is

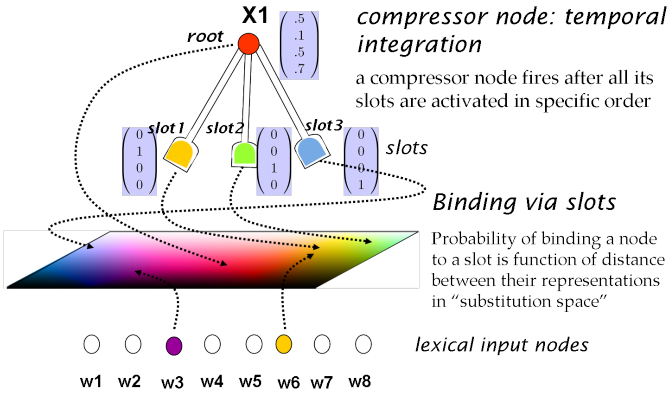


Figure 5.3: Enlarged image of a compressor node. The vector representation of the root determines its position in the high-dimensional substitution space (only 4 dimensions are shown).

a substitution operation, which is defined only between roots and slots (i.e., on $R_L \cup R_C \times S$), and only if $\langle r, s \rangle \neq 0$; one may also define *substitution probabilities*: $P_\circ(r \in R_L \cup R_C, s \in S) = f(\langle \mathbf{r}, \mathbf{s} \rangle)$.

5.2 Representational power of HPN

The representational power of HPN is at least equal to that of context free grammars (CFGs). It can be shown that the latter are a special case of an HPN grammar. This means that any CFG can be represented as a set of input nodes and compressor nodes in HPN with appropriate choices for their representations, i.e., by locating slots and roots at the appropriate positions in substitution space.[2] For the interested reader Appendix B.2 describes 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.

In this section I only give an example of a conversion from PCFG to probabilistic HPN. Table 5.1 shows a toy PCFG with relative clauses, and Figure 5.4, together with a specification of the vector representations in Table 5.2, shows the corresponding HPN grammar. By convention all slots are orthogonal, thus $S_1 = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$; $S_2 = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, etc. (the index i of slot S_i in Figure 5.4 indicates its only non-zero component.)

By defining a probability for binding a node to a slot one can assign probabilities to parses in HPN. The *binding probability* between a node and a slot is a

[2]The idea that the non-terminals of a CFG can be represented as points in a continuous vector space has been proposed before in Vector Space Grammars (VSG) [Stolcke, 1991]. With respect to their representation VSGs are similar to HPN, but otherwise (e.g., learning, combinatorial operations) the formalisms are quite different.

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

Table 5.1: Toy probabilistic context-free grammar with relative clauses (Adapted from [Elman, 1991]). Probabilities are indicated in brackets.

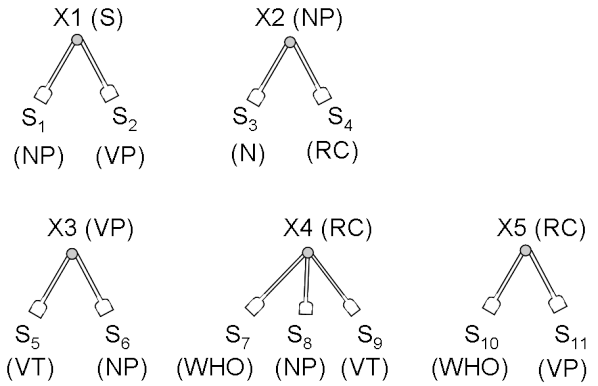


Figure 5.4: HPN grammar corresponding to the toy grammar of Table 5.1. Compressor nodes are labeled X1, X2, etc., and their vector representations are given in Table 5.2 (input nodes are not shown). The labels of the corresponding toy grammar are indicated within brackets.

Compressor nodes	
X1 (S)	-
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 5.2: Node representations of the HPN grammar depicted in Figure 5.4.

function of the inner product of their vector representations in substitution space (assuming vectors are normalized). For example, $P_{bind}(X2, S_1) = 0.3$.

The product of the probabilities of all bindings involved in an HPN derivation gives the HPN parse probability.The reader may verify that for any parse of a sentence produced by the artificial PCFG grammar of Figure 5.1 there exists a parse by the corresponding HPN grammar with the same branching structure and the same probability as assigned by the PCFG.

5.3 Parsing with the HPN grammar

A key feature of HPN, which distinguishes it from most conventional neural networks is an ability to assign a constituent structure to sentences. The component responsible for the network's ability to parse is a *dynamic binding* operation, which functions as a neural correlate of *substitution* (see section 2.4 for the motivation and section 5.7 for implementation details). Dynamic binding implies that *tags* are temporarily stored in the slots of compressor nodes and in the nodes that bind to them during sentence processing. A pair of identical tags implements a so-called *path connector*, functioning as a pointer from a slot to a node that has bound to the slot. An ordered sequence of path connectors stored in the local memories of the units constitutes a neural correlate of a *stack*. Whereas in symbolic parsers the stack is an abstract construct, that operates externally to the system, the HPN stack is distributed over the nodes that participate in a parse.

Using the path connectors, HPN can keep track of the trajectories of activation through the network as well as the branching decisions along the path. From the path connectors a *derivational path* can be reconstructed that corresponds to a parse tree. In HPN parsing to arbitrary many recursive levels and with arbitrary branching structure is possible. Figure 5.5 shows some typical parsing scenario's in HPN.

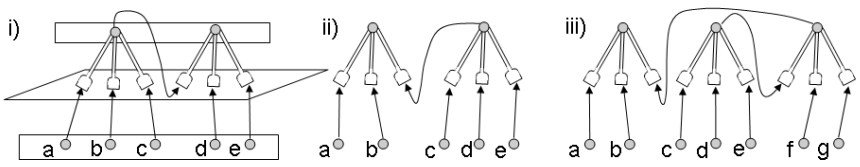


Figure 5.5: Scenario's of three HPN parses. i) ((a b c) d e), ii) (a b (c d e)), iii) (a b ((c d e) f g)). Path connectors are indicated by arrows.

Definition 2 (HPN parse and derivation)**.** A *parse* in HPN is a connected trajectory through the HPN network that binds a set of input and compressor nodes together through path connectors. An *HPN derivation* is an ordered sequence of steps through the HPN network, that fully determines an HPN parse, starting

from any compressor node, and every subsequent step obtained by application of the substitution operator. A parse of a sentence $x_1, x_2 \ldots x_n$ is successful if there exists a derivation such that

1. every successive word x_i in the sentence has its associated input node bound to a slot of a compressor node, while the order of the slots in every compressor node is respected.

2. every slot of every compressor node in the derivation 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. neighboring slots of a compressor node are bound to nodes that process adjoining parts of the sentence.

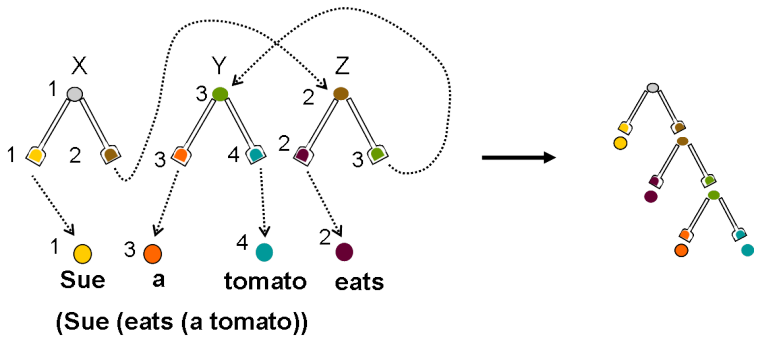


Figure 5.6: Derivation of the sentence *Sue eats a tomato*. The numbers correspond to the word position in the sentence. Dotted lines indicate bindings.

A (simplified) example of an HPN derivation is given in Table 5.3, and illustrated in Figure 5.6 for the sentence *Sue eats a tomato*.[3] The figure also shows the mapping between an HPN parse and a conventional parse tree.

X	$\rightarrow$	s1 s2	s1	$\otimes$	*Sue*
Y	$\rightarrow$	s3 s4	s5	$\otimes$	*eats*
Z	$\rightarrow$	s5 s6	s3	$\otimes$	*a*
s2	$\otimes$	Z	s4	$\otimes$	*tomato*
s6	$\otimes$	Y			

Table 5.3: A derivation of the sentence *Sue eats a sandwich*. Capital letters indicate compressor nodes; s1, s2, etc. indicate slots; *italics* indicate input nodes; $\otimes$ indicates a binding.

[3]As will be explained in section 5.3.2, the word position (indicated in the Figure) is one of two numbers that characterize the *state* of a node in the HPN network.

5.3.1 Probabilistic left corner parsing with HPN

HPN implements a left corner parsing (LCP) strategy to search for a syntactic analysis of a sentence. Left corner parsing [Rosenkrantz and Lewis II, 1970] is a cognitively plausible parsing strategy, because it proceeds incrementally from left to right, and combines top-down and bottom-up processing. This is compatible with a commonly held view of cortical processing and complex object classification in different modalities [e.g., Ullman, 2007, Hawkins and Blakeslee, 2004]. In this view, processing in the brain takes the form of an interaction between top-down hypothesis formation followed by prediction on the one hand, and bottom-up expectation matching of the hypothesis to the input signal on the other hand.

As a matter of fact, top-down parsing is not an option in HPN, neither is strictly bottom-up parsing, because in both strategies the search would never terminate, as in a typical initial set-up every slot of an HPN compressor node can bind to the root of any other node. Moreover, since there are no labels in HPN (or for that matter in the brain) there does not exist a priviliged start category (i.e., S), from which to start the top-down parsing process; in HPN a derivation can start from (and a parse can be rooted by) any arbitrary compressor node. In left corner parsing on the other hand, the search process is initiated by bottom-up activation of words, and constrained by top-down derivation of goal categories, while recursive application of a rule is limited to one step.

As was explained in section 3.1.3, in LCP a derivation of a sentence involves three parse operations: *shift*, *attach* and *project*. The difference between HPN parsing and symbolic left corner parsing is that in symbolic left corner parsing one compares a left corner category to a goal category, whereas in HPN one tries to match the root of a (compressor or input) node to a left slot in case of project, or to a non-left active slot (called a *goal slot*) in case of attach. Further, symbolic left corner parsing starts a parse with the initial state S as a goal category, whereas in HPN any compressor node can initiate a parse. Appendix B.1 gives the pseudo-code of an algorithm for deterministic and serial parsing in HPN.

In the experiments of this chapter I have used a probabilistic version of this algorithm to compute the *most probable parse* of a sentence. In this case binding is not all-or-nothing, but it has a certain *binding probability* that can be computed from the metric (the topology). The conditional probability of binding the root of a particular node to a slot, given the root, is given by the inner product of the root vector and the slot vector, normalized by summing over the inner products of the root vector with all slot vectors. The probability of a parse can then be calculated as the product of the binding probabilities involved in an HPN derivation.

5.3.2 HPN node states

In order to keep track of multiple trajectories through the network that run in parallel (possibly crossing the same node multiple times), and to track back a

trajectory, HPN must distinguish the particular state of a node that participates in the parse. A *node state* refers to a specific instantiation of a compressor or input node in a parse, where multiple instantiations involving the same node within a single parse are possible. In analogy to states in a Earley parser [Earley, 1970] (see section 3.1.11), in HPN the state of a compressor or input node is characterized by means of its unique root X, its active slot (compare the position of the dot in an Earley state) and 2 additional numbers, i and j. In HPN j is called the rootIndex, that is the index number given to the production root when it was first activated; i is called the slotIndex associated with the active slot. With the formal definition of a node state it is possible to make some of the concepts that were introduced earlier more precise

Definition 3 (HPN node state)**.** An HPN node state $\mathcal{S}$, associated with a (input or compressor) node X, is a 4-tuple $\{X, n, i, j\}$, where X uniquely identifies the root of the node, j is the root index, i is the slot index associated with the active slot, and n is the ordinal number of the active slot in the production (the leftmost *unbound* (free) slot that is expecting bottom-up input).

Definition 4 (Completion)**.** A node state is called *complete* if all slots associated with the node have been bound. This is annotated by $\{X, c, i, j\}$. A state that is not complete is called *incomplete*. The state of input nodes is complete by default. (Given an input node W representing a word w, and given that word w occurs at position i in the sentence, then by convention the state of the input node is $\{W, c, i, i + 1\}$.)

Definition 5 (Path connector)**.** Given node states $\mathcal{S}$, the set of incomplete node states, I, and the set of complete node states, C, then a *path connector* $\mathcal{P}$ is a function (or pointer) $\mathcal{P} : \mathcal{S}_i \in I \rightarrow \mathcal{S}_c \in C$ from an incomplete to a complete node state.

Definition 6 (Substitution operator)**.** The *substitution operator* Σ is a function $\Sigma : (C \times I) \cup C \rightarrow I \times \mathcal{P}$. In words: the substitution operator is a function from a complete state (in case of projection) or from a complete plus an incomplete state (in case of attachment), that creates a new incomplete state plus a path connector from the new state to the complete state.

5.4 Learning

While thus far it has been shown that a grammar can be *represented* in HPN (e.g., see section 5.2), the added value of the connectionist approach is in its ability to actually *learn* syntactic representations. In HPN parsing and learning are complementary, as node representations are adjusted after every parse. Although I have not yet discussed the implementation of a parser for HPN, we may assume

for now that an optimal parse has been found and the bindings between nodes are slots are available for learning.

In order to induce a meaningful topology, HPN must somehow induce the vector representations of the nodes from the distribution implicit in the corpus. Like the BMM algorithm [Stolcke and Omohundro, 1994] (see section 3.1.11) HPN concludes that two nodes are syntactically (paradigmatically) related when they appear in similar contexts (i.e., in the same slot). However, unlike the discrete merging step of BMM, HPN makes the nodes more substitutable in a gradual way, by decreasing their distance in substitution space upon encountering similar contexts. 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 in section 5.2). Recover the productions and bindings involved, using the path connectors.
3. Learning. For every binding in the most probable derivation, move the bound node and slot closer to each other in substitution space. Adjust the representation of the node $\mathbf{n}$ that is bound to slot $\mathbf{s}$ according to $\Delta\mathbf{n} = \lambda \times \mathbf{s}$, with (decreasing) learning rate λ. Also adjust the representation of the bound slot $\mathbf{s}$ according to $\Delta\mathbf{s} = \lambda \times \mathbf{n}$.[4] (Alternatively, one may adjust the representations of the nodes in the neighborhood h of the 'winning nodes', in proportion to their distance in substitution space.[5])

As can be inferred from the learning step, the internal representations of words and syntactic categories are gradually changed from idiosyncratic (i.e., not correlated with other node representations) to systematic (and abstract), when they participate in more slots. Two input nodes that often participate in the same slot(s) will be gradually merged into a single part of speech category (see the example in the next section). 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.

[4]These update rules are compatible with an inner product metric. If instead Euclidean distances are used to compute binding probabilities, the update rules should be defined accordingly, thus $\Delta\mathbf{n} = \lambda \times (\mathbf{n} - \mathbf{s})$.

[5]Note that since units in HPN do not exist on a 'grid', one cannot define a 'physical' neighborhood as in the Kohonen learning algorithm. Instead of updating the physical neighbors of the winning node one must update its neighbors in weight space. In this respect the topology of HPN is more related to a so-called *neural gas* [Martinetz and Schulten, 1991], which is a variation of the Kohonen network without a grid.

5.4.1 An example of the formation of abstract categories through self-organization of the topology

Let us assume the HPN network is initialized with lexical nodes corresponding to the words *dog*, *cat*, *feed* and *the*, each having a random representation in substitution space, and a number of compressor nodes with random representations of their roots (see Figure 5.7). Suppose we then parse a corpus consisting of the two sentences *feed the dog* and *feed the cat*. What will happen is that the parse-and-learn cycle will cause the nodes for *dog* and *cat* to become related through self-organization of the topology, reflecting the corpus distribution. The reason is that if two words occur in the same contexts they tend to bind to the same slots. For instance, after parsing the first sentence, *feed the dog*, the two left slots of compressor node $X1$ are moved towards *feed* respectively *the* in substitution space. As a result, the second sentence, *feed the cat*, has a higher than random probability to be parsed using the same compressor node $X1$ (because the parse probability is the sum of the binding probabilities, which is a function of the distance in substitution space). Subsequently, if $X1$ is selected again as the preferred compressor node to parse *feed the cat*, the learning step causes the node for *cat* to be moved closer to the third slot of c-node $X1$, which itself was moved closer to the node for *dog* in the previous sentence. Thus, the slots mediate the formation of abstract category regions in substitution space through a process of 'contamination'.

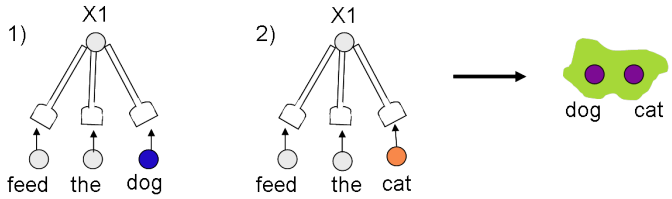


Figure 5.7: The formation of a shared category for *cat* and *dog* induced by distribution.

This shows that within the HPN framework a mechanism exists that can explain in principle the formation of constructions with abstract *slots* or fillers (e.g., *I want X*) (whether HPN is also successful in practice is the topic of the next section). In Usage Based Grammar [Tomasello, 2001] a similar process is proposed as an explanation for how syntactic categories are gradually bootstrapped, during the second stage of children's linguistic development.

5.5 Experimental evaluation

This section describes a number of experiments that aim to demonstrate the gradual formation of a topology of syntactic categories by HPN, and to evaluate

whether HPN discovers clusters of word categories from an artificial corpus, as in [Elman, 1991]. (For a more direct comparison of HPN with the SRN on the *word prediction* task it will be needed to develop a full probabilistic version of an HPN chart parser, that can compute so-called *prefix probabilities* (see the discussion in section 5.6.3). This will be the subject of subsequent chapters.)

5.5.1 Topology formation in an artificial language

In the first experiment HPN was trained on 1000 distinct sentences, including up to 5 levels of center embedding, which were generated at random from the artificial context free grammar given in Table 5.1 (a simplified version of the grammar with relative clauses used in [Elman, 1991]). The HPN network had 10 compressor nodes with 2 slots, and 5 with 3 slots, and all compressor and word node representations had 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 5.8 shows the representations of the input nodes after learning has completed (scaled to two dimensions using Matlab's `cmdscale`, i.e., multi-dimensional scaling). Also shown are the average compressor node representations that map to the symbolic labels of the gold standard parses (NP, VP, etc.). From the figure it can be seen that

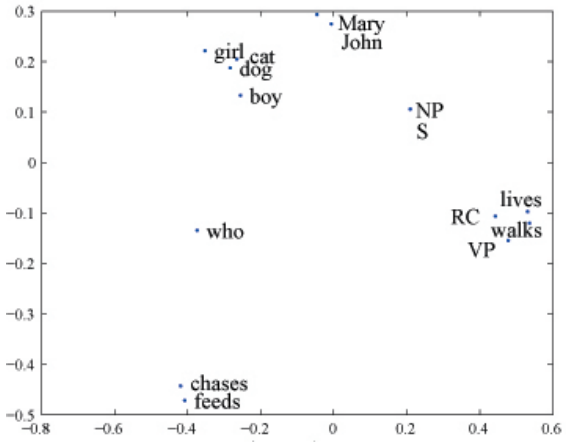


Figure 5.8: Substitution space with HPN node representations

clusters are discovered that regroup the words into their original categories: transitive verbs (feeds, chases) well separated from intransitive verbs (walks, lives), and nouns are distinguished from proper nouns. The clustering is very stable over many trials of the experiment, with different random initializations of the word and compressor node vectors. Moreover, some of the higher order categories are mapped in substitution space between words with which they are intuitively substituted. This indicates that HPN can in fact learn word categories from distributional information implicit in the random sentences.

5.5.2 Recursive systematicity

The second experiment attempted to evaluate recursive generalization and systematicity of HPN. Again, 1000 sentences were generated at random from the same recursive artificial context free grammar, but this time they were split into 800 train sentences and 200 test sentences that did not occur among the train sentences. To test whether the induced HPN grammar had indeed learned the intended grammar, and was able to generalize we let the trained HPN grammar parse the test sentences, and evaluated unlabeled precision (UP) and recall (UR) of the found constituents (see section 3.1.6).

In one variant of this experiment training was semi-supervised: the Gold standard brackets (but not the labels) were provided during training, thus constraining the possible parses of train sentences. Subsequently unlabeled precision and recall was evaluated of 200 novel test sentences which were supplied without brackets. The resulting scores of UP=UR=86,4% (similar scores were obtained in additional trials) indicate that the induced HPN grammar approximates the original grammar to a fair degree, and generalizes to the test sentences.

By contrast, in an unsupervised training variant, where no bracket information was supplied during training, unlabeled recall and precision on the same 200 test sentences varied between 40% and 80%, and was highly unstable. In this case it seems that HPN does not always converge to the original grammar, as was expected. This negative result leads to an interesting insight: apparently very many HPN grammars other than the original artificial grammar are compatible with the training sentences, and most of them do not generalize as expected. The train sentences by themselves thus impose insufficient constraints to induce a grammar that produces the intuitively correct systematic generalizations. This is a fundamental problem, to which I will come back in section 5.6, and it will finally be addressed in Chapter 8.

5.5.3 Topology formation in realistic corpora of children's speech

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. A demonstration of the emergence of a topology of word categories is given for the Eve corpus [Brown, 1973] from the CHILDES database MacWhinney [2000]. Figure 5.9 shows the results of one trial on 2000 consecutive child utterances from the second half of the Eve corpus, which were presented to the network together with the brackets available from the dependency annotation. The interesting clusters are accentuated. It should be noted though that subsequent trials did not yield comparable clusters.

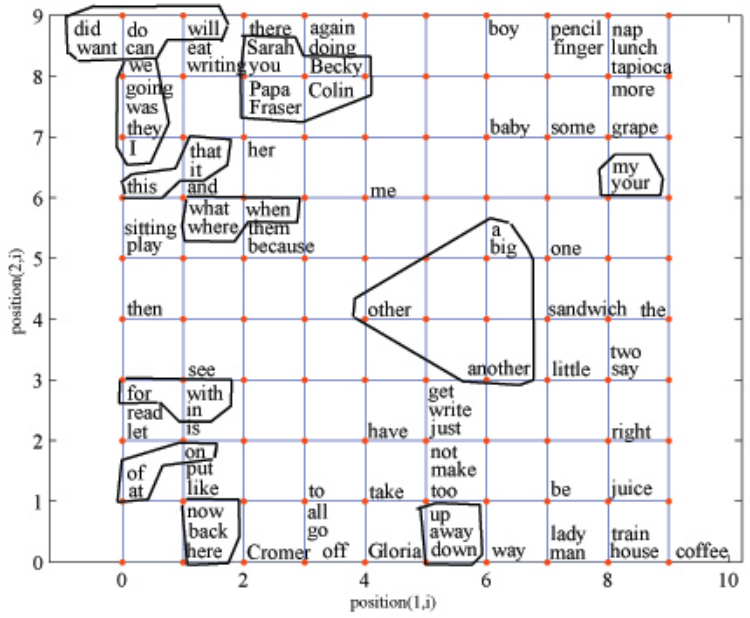


Figure 5.9: Representations of Eve's 100 most frequent words

5.6 Chapter conclusions

5.6.1 HPN and systematicity

By presenting a working implementation of a connectionist model that satisfies the systematicity requirements of language, this chapter achieved one of the aims set at the start of this thesis. For completeness let me briefly reiterate how HPN satisfies, by design, the two criteria of systematicity formulated in section 4.5.2.

- **The context invariance criterion**
  Recall from section 4.8.1 that a solution for creating context invariance in connectionist networks may take the form of *encapsulation* of the network units. As was explained there, this means that connectivity, or binding between units is determined by some 'intrinsic' representation of the units, which is independent of literal preceding context or incoming activation. In HPN the output of a compressor node (i.e., the root vector) is encapsulated (i.e., separated) from the input to the slots. Hence, the predictive behavior of a HPN unit depends on its identity alone, as required by the context invariance criterion in section 4.5.2.

  The slots of HPN compressor nodes fulfill the role of 'variables', i.e., they function as placeholders where units are bound. More precisely, they are *invariants*, because they have no predetermined or innate meaning (see section 4.8.1): the extensional meaning of a slot (i.e., its topological position

in substitution space) is continuously updated from experience. Thus, any slot in HPN can assume the role of an arbitrary non-terminal.

- **The recursive systematicity criterion**
  Recursive systematicity is satisfied in HPN by virtue of complex primitive network units, the compressor nodes. These perform hierarchical temporal compression, and thereby implement a 'context free' inductive bias (see also section 4.8.3). Whereas the compressor nodes take care of the syntagmatic (precedence) relations of the grammar, the topology, or substitution space encodes the paradigmatic relations of the grammar.

5.6.2 HPN versus RAAM

In Chapter 4 I mentioned RAAM [Pollack, 1988] as another instance of a connectionist model that explicitly encodes hierarchical structure. RAAM therefore implements a context free inductive bias as well. It is worth pointing out here the difference between HPN and RAAM. As was argued in section 4.3.1, for encoding and decoding a parse tree RAAM crucially relies on an external interpretation of the output of the hidden layer as either the 'left child' or the 'right child' of the current branch in the tree as it selects to which input array (left or right) to feedback the output of the hidden layer to at any particular level of embedding. It was argued in the same section that this makes RAAM a symbolic system.

By contrast, in HPN the decision whether to bind the root of a node to either a left or right slot or a compressor node is made autonomously and wholly unsupervised by the system, as determined by the topology. In other words, by contrast to RAAM, in HPN the stack is handled internally. Moreover, the topology is *learned* rather than prespecified. Further, in HPN all information necessary to 'read out' the branching structure of a parse is local and internal to the system (no external interpretations are required).

A second important difference, mentioned also in section 4.3.2, is that parse trees are represented in RAAM (and likewise in the tensor products of [Smolensky and Legendre, 2006]) on a case by case basis, by memorizing 'conjunctive', or 'static' bindings' between the constituents of the parse. Such static bindings have to be specified in advance for every sentence, hence are not productive and do not generalize [Hummel et al., 2004]. HPN on the other hand employs flexible, dynamic bindings between units, which allows HPN to encode (elements of) a *grammar* in the network units rather than a fixed set of parse trees. As a result of dynamic binding parse trees in HPN have an internal structure, unlike conjunctively encoded parse trees, allowing for relational inference and generalization to novel sentences.

5.6.3 Reaching the limits of HPN

Despite the fact that HPN is suitable for systematic language processing, it is still not a very good language model. The empirical tests of section 5.5 identified several limitations of the current version of HPN.

1. HPN is context free. As noted in section 5.2, the metric of HPN encodes context free rules, whose expansion probabilities (encoded as distances in the metric) are conditioned on local information alone, and which are insensitive to sentence history or structural context. However, in computational linguistics it is known that the regularities of natural languages cannot be captured by the context independence assumptions of the PCFG alone (see section 3.1.8); parsing accuracy generally increases when additional structural and lexical context is considered.

   Left corner parsers, as an instance of *history-based parsers*, make use of this insight by conditioning every parser move on a history of previous parser moves, thereby relaxing the independence assumptions of the PCFG. For instance the model of [Manning and Carpenter, 1997], discussed in section 3.1.7, conditions the probability of a parser move on both the left corner and a goal category that has been derived earlier in the derivation history.

   Since HPN also makes use of a left corner parsing strategy, it would be desirable that HPN could similarly condition its decisions on earlier parser moves. However, if the decision process of the system depends exclusively on the metric this precludes the incorporation of non-local structural and lexical context, since the metric can only express binary relations between units (i.e., distances), whereas conditioning an event on a left corner and goal category is a ternary relation. A richer approach will be needed to encode the conditional events in the network.

2. HPN cannot compute string probabilities. The left corner probability model of HPN, based on [Manning and Carpenter, 1997], can compute parse probabilities given a sentence, but not sentence or string probabilities. Yet, there are many interesting tasks other than parsing, such as the next word prediction task, that require an ability to calculate sentence or string probabilities. For a direct comparison of HPN with the SRN on the word prediction task HPN must be able to compute so-called prefix probabilities. For this purpose it will be needed to include and store information about shift transitions (from a compressor node to a word node) in the network (without violating the connectionist constraints) and thus turn HPN into a language model.

3. Learning generalizations by analogy. As the test of recursive systematicity of section 5.5.2 revealed, the fact that the train set consists of sentences

from a certain (recursive) grammar is by itself not a sufficient condition for HPN to learn that grammar. Apparently there are many possible HPN grammars compatible with the train sentences, and HPN is not guaranteed to converge to a 'correct' grammar, i.e., one that makes the same systematic generalizations as the original artificial grammar. What seems to be missing is some extra constraint that drives learning in the right direction. As was argued in section 4.5.3, to learn the correct structure of a grammar a learning algorithm must rely on finding structural analogies between previously processed sentences. This ability requires that the system can somehow access exemplars (parses) of previously processed sentences from memory. The question is how this can be implemented in a connectionist network?

Conditioning in neural networks: towards exemplar-based connectionism

The above issues, concerning the limitations of HPN as a model of natural language processing, raise one particularly important question: how can conditioning events be accommodated for in a neural network? In other words, how can a neural network encode the dependency of natural language on contextual history, while at the same time preserving some measure of context invariance to enable generalizations? Whereas in symbolic models of language conditional probabilities can simply be stored in an external look-up table, in a neural network this is not an option: the conditioning events must be locally accessible, since the parse decisions must be computed at a local level. On the other extreme, recurrent networks such as the SRN do take all previous sentence context into account, but, as we saw, cannot make systematic, context invariant generalizations. The ideal connectionist model of natural language must combine the best of both worlds.

Such considerations deem it necessary to store the analyses of previously processed sentences somehow in the network, in a way that allows utilizing these events for the analysis of novel sentences. I propose a neurally plausible solution for conditioning on sentence context in neural networks: the use of episodic memory. The assimilation of an episodic memory for sentences in the network allows moving from a purely *rule-based* connectionist approach to language processing (the current version of HPN) to an approach that integrates abstract, rule-based linguistic knowledge with *exemplar-based* knowledge.

In the next chapters I will introduce an exemplar-based version of HPN, *episodic-HPN*, in which the HPN network is extended with an episodic memory — a memory for concrete instances of processed sentences. From now on, one should see the version of HPN presented in this chapter as a model of semantic memory, which is a memory for abstract, rule-based knowledge, that is but one component of a larger system that integrates semantic and episodic memory.

5.7 Neural interpretation of core components of the HPN model

Before adding additional complexity to the model in the following chapters, I conclude this chapter by asking whether the components of the current model can be implemented in neural hardware. While some of the proposed ideas are necessarily still rather speculative in nature, they are in part also intended to show the compatibility of the notions introduced in this chapter (e.g., dynamic binding, node states) with a neural design, and with the connectionist constraint of locality.

The substitution space

Unlike topologies in the visual or auditory cortex, the topology of the substitution space is implemented in a 'virtual' space. This means that columns representing neighboring vectors in substitution space are in general not physical neighbors on the cortical surface, but may be located in remote regions of the cortex.

For example, in Figure 4.11 columns for *dog* and *fog* or *cat*, *can* and *cap* are physical neighbors in a (hypothetical) topology in auditory cortex, but at the same time I claim that the lexical representations of *dog* and *cat* cluster in a topology in a 'conceptual' space. This is a quite bold claim that deserves some (neurally plausible) clarification.

For the clarification and neural motivation of the substitution space I refer back to section 4.8.2. One part of the answer is that 'conceptual' representations are encoded in the 'name fields' of cortical columns, as proposed by Hawkins and Blakeslee [2004, p.150-2] and mentioned in section 2.1.3 of Chapter 2. These are intrinsic representations, presumably stored in cells in L3 of a column, that link a column by its name (i.e., 'by reference') to columns in higher cortical levels, *independently* from bottom-up neural activity.

The second part of the answer is that these intrinsic representations, or 'names' are linked in a 'virtual' topology through a switchboard construction, as explained in section 4.8.2. Thus, while columns representing syntactic categories may be located in diverse and physically remote areas in the cortex, by virtue of the switchboard their intrinsic representations are unified in a coherent, *virtual* topology. This topology corresponds to the substitution space, and it is learned by updating the column representations across bindings (as explained in section 5.4).

5.7.1 Connectionist implementation of a switchboard

Figure 5.10 illustrates how the switchboard could be implemented in HPN (but note that the current implementation of HPN abstracts over most of these details). Incoming connections to the switchboard originate from the root of a compressor

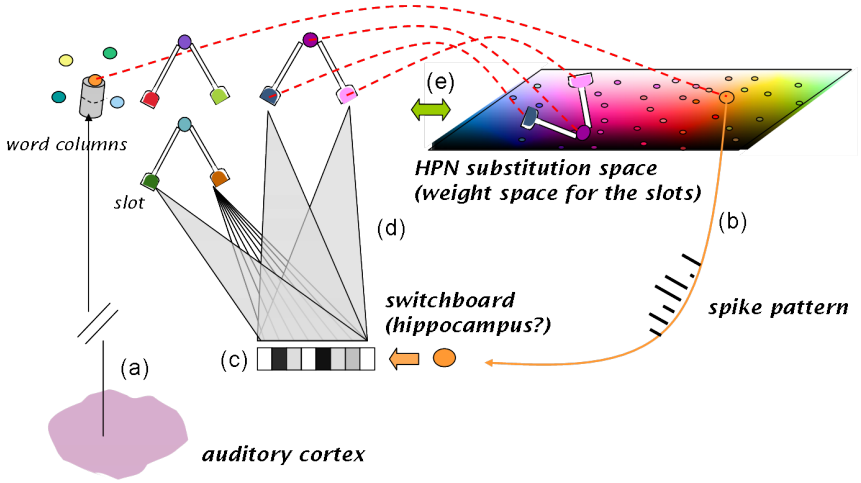


Figure 5.10: Schematic illustration of the substitution space and switchboard mechanism for dynamic binding. a) The word units are directly bottom-up activated from the auditory cortex (not modeled in HPN), while the slot units are activated via the switchboard. b) An activated unit in substitution space transmits its virtual topological address to the switchboard. c) The address is encoded as a distributed pattern on the switchboard units, which project back to slot units in the substitution space (competitive layer). d) The switchboard units are fully connected to all slot units (illustrated schematically for only 4 slots). e) The weight vectors of the slots and the concept-side coordinates of columns correspond to positions in substitution space (dotted lines).

node or a word unit, and are transmitted serially as a spike pattern, while outgoing connections attach (physically) to all slots.

In order to match an incoming signal to a slot, the switchboard transcribes the spike pattern (b) (which encodes the 'name' of a unit, corresponding to its address) on a set of switchboard units (c). The latter are fully connected to all slots (d). (Hence, the number of weighted connections to each slot equals the number of switchboard units, and this in turn determines the dimension of the substitution space.) The substitution space can be thought of as a competitive, self-organizing layer [Kohonen, 1998], whose units compete for the best match with the activation presented on the switchboard units. (Note from Figure 5.10 (a) that it is assumed that word columns are activated from bottom-up through their perceptual pole, but how this is implemented falls outside the scope of the HPN model, since HPN only models top-down processing.)

5.7.2 The tagging system

While the switchboard mechanism described above demonstrates the capacity for flexible binding, it still lacks the ability to keep bound units together for some time (in working memory). The ability to temporarily store bindings in working memory is an essential requirement for comprehending complex propositions. Specifically, in case of syntactic processing the ability to reactivate a binding at a later point in the derivation is critical (e.g., for back tracking). However, in order to trace back a previously bound unit it is not sufficient to return to its known topological position via the switchboard. For instance, in HPN two compressor node root vectors (of which one has 2 slots and the other 3 slots) may occupy the exact same position in the topology, yet it is needed to distinguish which of them is bound to a certain slot in a pending derivation.

For dynamic binding to work it is thus required that some *tagging mechanism* is in place, that adds an explicit and unique *tag* to group units that are bound (e.g., a slot with the root of a compressor node in HPN). (The same point is also made in the work of [Hummel and Holyoak, 1997, p.7] on modeling analogical inference, in the context of an interesting discussion on dynamic binding.[6]) Thus, there are two necessary and dissociable components to dynamic binding, which

[6]In contrast to HPN, in the LISA model for analogical inference [Hummel and Holyoak, 1997, 2003] it is assumed that dynamic binding is implemented by grouping units that fire in synchrony [Singer and Gray, 1995]. In the binding by synchrony view the neural substrate of a tag is the oscillation 'phase' (e.g., within the Theta frequency range). Binding by synchrony is however problematic, for reasons discussed in section 2.3. Hence, in this work I have opted for serial dynamic binding. In that case the tag consist of some neural substance that is shared among bound units, as explained in the main text. Serial binding solves one of the challenges that Hummel and Holyoak [1997] have acknowledged the synchronous binding approach cannot deal with, namely how to save the bindings to long term memory. In Chapter 8 I will argue that the tagging mechanism proposed here for serial binding is also used by the brain to store long term episodic memories.

have complementary functions: an 'addressor system' (the switchboard), responsible for flexible creation of new bindings, and a 'tagging system', responsible for maintaining existing bindings.

Together, the address (encoded as 'name' vector in a column) and a stored tag provide sufficient information to reconstruct the exact original binding, and eventually the derivation: the address defines a local neighborhood in substitution space where the bound slot can be found (via the switchboard); subsequently, the tag allows for identifying the exact slot. Thus, the topological address plus tag together implement a *path connector* (i.e., a pointer), as defined in section 5.1. In Chapter 8 I will argue that the same 'address plus tag' mechanism is also used by the brain to replay episodic memories.

The fact that tags are used for syntactic processing imposes some minimal conditions on a potential neural correlate of a tag. For instance, it is required that tags are able to distinguish the bindings of a pending derivation, and that they somehow encode the (hierarchical) order of the bindings as the derivation progresses through the sentence.

Local neuronal short-term memories

The dynamic binding and tagging mechanism assumes that while processing an event (e.g., during parsing) tags are temporarily and locally stored in cortical columns. As mentioned in section 2.3, there is indeed evidence that the brain stores a reference to recent bindings in local short-term memories at the level of neural assemblies. In particular, it has been found that neurons sustain their *presynaptic* connections through calcium-mediated synaptic facilitation for periods of around a second [Mongillo et al., 2008, Barak and Tsodyks, 2007]. This is about the amount of time needed for the path connectors to keep a tag alive that identifies a bound node (assuming they are used for sentence processing).

Barak and Tsodyks [2007] claim that transient presynaptic facilitation is a neural correlate of working memory. In line with this view, in HPN an ordered set of path connectors (realized by numbered tags) constitutes a distributed *stack*, which is essentially the working memory of a parser. Both calcium-mediated presynaptic facilitation and the interpretation of a stack in HPN are consistent with contemporary theories in cognitive neuroscience, according to which working memory is not localized in any particular module, but is a local *process* [Cowan, 1995].

Node states

A 'state' of a parser is a formal construct. Yet, the fact that formal theories of syntactic processing reason about states is a consequence of some underlying biological reality. I argue that states are concomitant to dynamic binding: the possession of a *tag* induces a state in a node of the network. (As argued before,

the tags must satisfy certain conditions that allow them to reconstruct the order of the derivation.) Thus, all interactions to keep track of the node states in HPN can be performed locally, and no 'symbolic operations' are required.

Neural instantiation of a compressor node

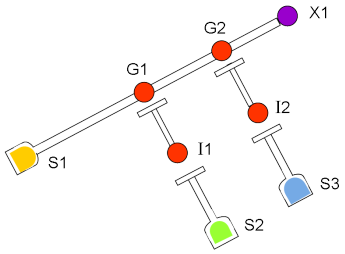


Figure 5.11: Hypothesized neuro-biological detail of a compressor node.

Figure 5.11 shows a neural configuration that can hypothetically implement the operation of a compressor node, as it forces the slots to be activated in the correct order. In Figure 5.11, S1, S2, and S3 are neurons that correspond to the slots of a compressor node, and X1 corresponds to its root. The 'gate neurons' G1 and G2 are controlled (gated) by two inhibitory neurons, I1 and I2, that fire by default (possibly driven by slot S1). When I1 or I2 fire they close the gates, and block the flow from S1 to X1. When the inhibitory neurons are in turn inhibited by the slots S2 and S3, the gates G1 respectively G2 are opened. When this happens in the correct order, activation can flow from S1 via S2 and S3 to X1.

Substitution space versus unification space

The HPN framework bears some similarity with a theory of neural syntactic processing that has been proposed by Vosse and Kempen [2000]. In their work, a 'unification space' fulfills more or less the same role as the substitution space in HPN. Vosse and Kempen [2000] propose that in the unification space lexicalized syntactic frames, which they borrow from Lexicalized Tree-Adjoining Grammars (LTAG) [Joshi, 2004], are unified into a parse of the sentence. Within the unification space syntactic frames compete among each other for alternative analyses of a sentence, via a complex lateral inhibition mechanism that involves temporal decay of unification strength. (In their original model though the syntactic frames are innate and symbolic, whereas an essential aspect of HPN is that it shows how the substitution space can be learned from scratch.)

Recently, functional neuro-imaging studies have tried to link the syntactic unification space with the left inferior frontal gyrus (LIFG) [e.g., Hagoort, 2005]. It has been suggested that the LIFG possesses all the crucial anatomical and

computational characteristics needed for the task of temporally integrating independent syntactic elements. In particular, the prefrontal cortex is well-suited for the maintenance and integration of information over time, and it is well-connected to the temporal lobe, where it is believed that lexical frames are stored. In section 8.4.2 I will however make the case that the substitution space of HPN can perhaps be identified with another brain structure, namely the hippocampus.

HPN and connectionism

It is a legitimate question to ask whether HPN is connectionist, as it obviously does not answer to the standard description of parallel processing and activation spreading networks. Yet, while traditional parallel distributed processing networks are perhaps suitable for modeling certain perceptual classification tasks (as these are characterized by massively *parallel*, *bottom-up* processing, and mediated by activation spreading through *conjunctive* bindings), many complex cognitive behaviors – among which language processing – require that the outcome of one subprocess is chained to the input of another process, that is such behaviors are processed serially and seem to involve some sort of binding [e.g., Lashley, 1951, Zylberberg et al., 2010]. Also in vision, "beyond a certain preprocessing stage, the analysis of visual information proceeds in a sequence of operations, each one applied to a selected location" [Koch and Ullman, 1985, p. 219].

To facilitate information exchange between local subprocesses, a more sophisticated means of communication between neurons is needed than can be provided by mean firing rates. Indeed, there is growing empirical evidence that neurons, or neuronal assemblies, utilize the potential of encoding rich information in precise spike-timing dynamics, and this has been followed up by an increasing interest among theoretical neuroscientists in the properties of artificial spiking neural networks [e.g., Izhikevich, 2006].

One should see the HPN network in the context of the countercurrent systems framework, mentioned in section 2.6.2. According to this framework cortical processing proceeds in two relatively independent networks, a bottom-up network, which is assiociated with parallel, bottom-up processing during a so-called 'feed forward sweep', and a top-down network, which is associated with recurrent, serial processing. As HPN is intended as a model of syntactic processing, which is a top-down and goal-driven task, the HPN model, unlike most other connectionist models of language processing, describes information processing in the *top-down* network.[7] The substitution space of HPN (together with the switchboard) can accordingly be considered as one possible formalization of the concept of a top-down network.

[7]To be entirely correct, sentence processing, or left corner parsing for that matter, is an interaction between bottom-up and top-down processes. However, the bottom-up processes, which correspond to activating the word assemblies for every word in the sentence (i.e., the input nodes in HPN), fall outside the scope of the HPN model.

Given the wide variety of solutions for neural processing in the cortex, and the many things that are still unknown about it, what then counts as a criterion for a model to be connectionist? As argued in section 4.2.1, I believe that the only criterion by which one can judge a model to be connectionist is what I have called the 'connectionist constraint'. This basically states that all operations (and interpretations) of the system should be limited to local network interactions. In this section I have therefore tried to show that the core components of the HPN model are all in principle compatible with the connectionist constraint, that is, they can be implemented using local interactions alone. In particular, I showed that the dynamic binding operation can be executed through local exchange of addresses (encoded in spike patterns) and tags between units , via a central switchboard. Thus, there exists a *connectionist* solution for referential operations, or 'pointers'. This allows HPN to implement a neural correlate of substitution, which is essential in order to account for the productivity of language.

Also left corner parsing, although not necessarily the algorithm employed by the brain, is not in principle incompatible with the connectionist constraint: in HPN it is assumed that network units have local short-term memories that function as registers. These registers keep a local queue of operations (i.e., shift, project and/or attach), and locally enforce their execution in the correct order (i.e., from the left to the right slot of a compressor node). Thus, while every compressor node and input node in the network functions as an autonomous processing unit, together they implement a left corner parsing strategy, without appealing to a central control mechanism.

Chapter 6

Episodic grammar

In this and the following chapters I will introduce episodic-HPN, an extension of HPN with an episodic memory. The model is based on an original hypothesis about the interaction of semantic and episodic memory in language processing. It shows how language processing can be understood in terms of memory retrieval, or as a priming effect, and language acquisition in terms of memory consolidation. I will point out that the perceived dichotomy between rule-based versus exemplar-based language modeling can be interpreted in a neuro-biological perspective in terms of the interaction between a semantic memory system that encodes linguistic knowledge in the form of abstract rules, and an episodic memory that stores concrete linguistic events. Before I present the full episodic-HPN model in Chapter 8, I will consider in this chapter the concept of parsing with an episodic memory for the supervised and symbolic case, using nonterminal labels learned from a treebank. I will implement a probabilistic, episodic grammar and evaluate its performance as a reranker on a realistic corpus of natural language, the Wall Street Journal.

6.1 Episodic memory

The previous chapter (section 5.6.3) identified several limitations of the current version of HPN. For instance, since all information available to the model are the

metric distances between network units, the model cannot represent (sentence) context, or do contextual conditioning. As a consequence, although HPN is suited for emulating (probabilistic) *context free* grammars (as was shown in section 5.2), it is ill-equipped for realistic language processing, where decisions depend on structural and lexical sentence context. The HPN network encodes, through the substitution space, context free relations between abstract (encapsulated) syntactic units, and as such qualifies as a *semantic* memory for the syntactic domain.

As defined in section 2.7, *semantic memory* refers to a person's general world knowledge, including language, in the form of abstract concepts that are systematically related to each other; *Episodic memory*, on the other hand, is a person's memory of personally experienced events or episodes, embedded in a temporal, spatial and emotional context (see section 2.7 for an extensive discussion of the human memory system).

The ideas developed in this chapter start from the observation that the scientific debate on the relation between semantic and episodic memory parallels, in a striking manner, an ongoing controversy about modeling language: one side in the debate is focusing on evidence for abstract, rule-based grammars [e.g. Marcus, 2001], and the other side emphasizes the item-based nature of grammar with a role (particularly in acquisition) for concrete sentence fragments larger than rules [e.g., Tomasello, 2000b]. While a rule-based grammar can be conceived of as an instance of semantic memory, as it encodes abstract, relational linguistic knowledge, the item-based approach suggests a role for episodic memory in sentence processing, since it reuses concrete (rather than abstract) linguistic experiences that have been memorized.

Assuming that the language domain mirrors cognitive processes from other domains, one expects that it would be illuminating to incorporate the notion of a semantic-episodic memory interaction within a computational model of language processing. In this chapter I will formulate a theory of episodic-semantic memory interaction, and based on this an 'episodic' model of syntax, called 'episodic grammar', that links language processing to memory processes, or more precisely to episodic memory retrieval. While the current chapter as a first step only introduces a *symbolic* episodic grammar – leaving out the topology – in Chapter 8 I will enrich the 'semantic' HPN model of the previous chapter with an episodic memory for sentences, thus lifting its limitation for dealing with sentence context. The episodic grammar model should take into account the following basic empirical facts about episodic memory

- *Physical traces.* All episodic experiences that occur during the lifespan of an individual, and that can be consciously remembered, leave physical *memory traces* in the brain. This includes memories of sentences that have been processed by the language system.
- *Chronological order preservation.* Most people are able to recover the ap-

proximate *chronological order* of their episodic memories. Thus, the relative order of the episodes must be somehow encoded in the representations of their traces.

- *Content addressability.* Priming effects demonstrate that static memories (for instance the memory of a smell) trigger episodic memories that are strongly associated with them. It is commonly believed that retrieval of episodic memories is contingent on cues from semantic memory. To account for *content addressability* an episodic memory must support local access from semantic memory units to their associated episodes (as implemented for instance in Hopfield networks [e.g., Hopfield, 1982]).
- *Sequentiality.* In the Memory Prediction Framework it is emphasized that the function of memory is to make temporal inferences (i.e., predict). According to [e.g., Eichenbaum, 2004] episodes are construed as temporal sequences of (time-less) semantic elements, bound together within a certain context (see Figure 2.7 in section 2.7.1).
- *Separatibility and identifiability.* The memory system must be able to identify and disambiguate an episode, even if it overlaps with another episode that is partly composed of the same semantic units. It is thought that to this end special 'context neurons' exist, that fire only for the duration of a specific episode [e.g., Levy, 1996].

There exist several connectionist models of episodic memory in the literature [e.g., Hopfield, 1982, Miikkulainen, 1999, McQueen, 2005, McClelland et al., 1995]. In section 8.5.1 I will discuss an instantiation of the latter, [O'Reilly and Norman, 2002] in the context of memory consolidation. Yet, as far as I know, to date there exists no theory of episodic-semantic memory interaction that is applicable to syntactic processing.

6.1.1 Proposal for the representation of episodes as distributed traces in semantic units

I propose that the episodic memory of a sentence is distributed across semantic memory units (i.e., the HPN nodes), and consists of physical traces, contained inside the nodes, that keep a record of the nodes participation in the derivation of the processed sentence. (In general, I claim that the episodic memory of a complex event consists of physical traces, distributed across the primitive semantic units that took part in structurally encoding the particular event.) This is illustrated schematically in Figure 6.1, which shows the episodic memory traces in the HPN network after hearing the sentences *girl who dances likes tango* (light colored traces) and *boy likes mango* (dark colored traces). Each of the traces of a processed sentence points to its succeeding and preceding node in the derivation, which allows HPN to reconstruct the original derivation from the traces. Conceptually, all that is required to upgrade from semantic HPN to episodic HPN is to

turn the existing local *short term memories* in the slots, where pointers to bound nodes are temporarily stored, into *long term memories* after a sentence has been successfully processed. (Note that this proposal implies that the semantic units involved in encoding an episode, like those involved in an HPN derivation, are *dynamically* bound: see section 8.4.2 for a neural perspective on episodic memory encoding in HPN.)

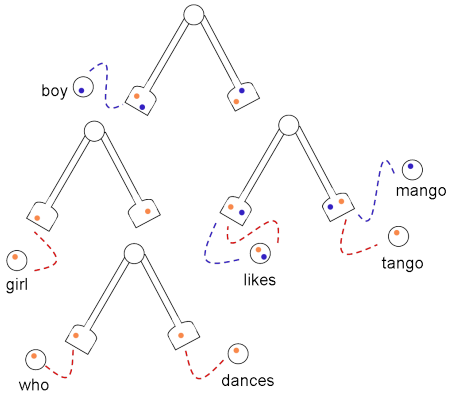


Figure 6.1: Episodic traces of a sentence (drawn as colored dots) are stored in local memories of visited nodes in the HPN network. In HPN the nodes and slots are situated in a topology.

6.2 Episodic grammar — model outline

For a more formal introduction to the topic of episodic grammar, let us leave for the moment the framework of HPN, and first deal with a symbolic implementation of episodic grammar. I will come back to the HPN formalism in Chapter 8, when I will work out the details of episodic-HPN. Also in the symbolic approach it is useful to take the point of view of a grammar as a network of interconnected *treelets*, that can combine with each other through substitution. I will assume that context-free rules from traditional grammars correspond one-to-one to such treelets, which thus play the role of the *compressor nodes* in HPN. I will also assume that the treelets possess a register (an internal memory, corresponding to a *slot* in HPN) that keeps track of the correct order of application of the syntactic operations.

As in HPN, in the episodic grammar a derivation is a sequence of visits to treelets, whereby treelets are bound through serial binding. The standard approach assumes a *top-down*, left-to-right derivation: each next rule is combined through left-most substitution with the partial tree derived so-far. I will also consider *left-corner derivations* in the next section.

In order to remember the correct order of derivation (which can vary depending on the chosen derivation strategy) the episodic traces encode the sentence

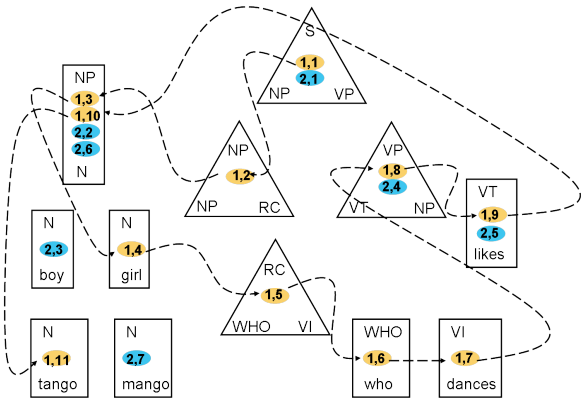


Figure 6.2: Episodic traces of two sentences (drawn as colored ovals) are stored in local memories of visited treelets (indicated by triangles and rectangles) in the *symbolic* episodic network. Note that by virtue of their ordinal number the traces implement pointers to successor treelets in a derivation (drawn for the first sentence alone).

number (s) as well as the position (k) of the treelet within the derivation. In Figure 6.2 the traces (for a top-down derivation) are identified by these two numbers, indicated as $\langle s, k \rangle$ inside the treelets. Note that after hearing many sentences a single treelet will store traces for all sentences that have visited it, which are distinguished by their sentence number, and possibly multiple visits from the same sentence.

The episodic sentence memories stored in the traces can also be recruited for the purpose of processing novel, unseen sentences. The idea is that when the derivation of a novel sentence arrives at a treelet, the traces encountered within the treelet trigger memories of stored exemplars. These receive an activation value whose strength depends on how close the stored derivation is to the pending derivation (see section 6.2.3). Every next step in the derivation is determined by competition between traces of different exemplars, each having its own preference for a successor treelet, and its own activation strength. In this view sentence processing (or parsing) can be interpreted as being subject to a *priming* effect: the traces prime or reactivate derivations of previously processed sentences (through content addressability), and restore the memory of previous parser decisions.

The above proposal satisfies the requirements of an episodic memory, as mentioned in the previous section, and it conforms to the view that episodes consist of pointers that bind semantic memory units into temporal sequences [e.g., Shastri, 2002]; content addressability is satisfied because activation of a single trace in a semantic unit triggers an entire episode. Parts of episodes are thus *reconstructed* on-the-fly at test time, rather than searched for. Further, chronological order preservation, as well as sequentiality and separatibility are trivially satisfied by the way that traces are encoded. Given a probabilistic interpretation, the episodic

grammar offers an explicit computational instantiation of the reinstatement hypothesis of episodic retrieval.

6.2.1 The left corner episodic grammar

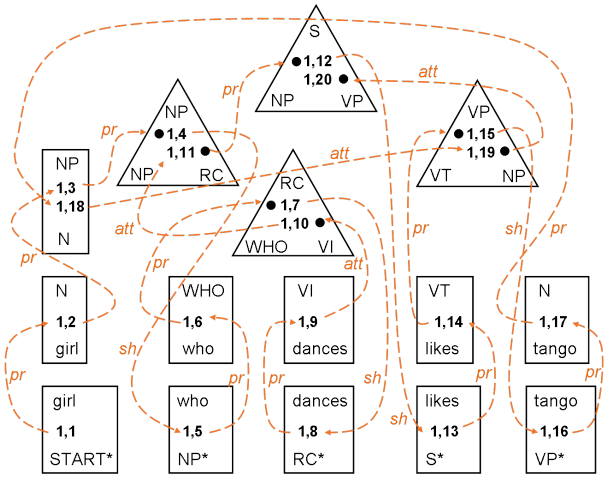


Figure 6.3: Episodic memory traces in the left corner episodic grammar after deriving the sentence *girl who dances likes tango*.

One of the advantages of the episodic approach is that it allows for comparing different derivation strategies within a single framework, and find out what the effect is of a different order of application of operations on treelets. An interesting parsing strategy from a cognitive point of view is *left corner parsing* [Rosenkrantz and Lewis II, 1970], since it proceeds incrementally from left to right, and combines top-down and bottom-up processing.

As explained in section 3.1.3, in left corner parsing the grammar rules are introduced bottom-up by a *project* operation to the *left corner* of the rule. The 'left corner' is the left-most symbol on the right hand side of a phrase structure rule; in the episodic framework it refers to the bottom-left nonterminal of a treelet. As long as there are no *completed* (i.e., fully processed) treelets, the next word in the sentence is introduced by a *shift* operation; otherwise the derivation can either *project* to a new treelet, or *attach* to a not yet completed treelet that has been previously introduced. Figure 6.3 shows an episodic left corner derivation for the sentence *girl who dances likes tango*. The shift, project and attach operations are indicated in the figure by their abbreviations.

Whereas most standard probabilistic left corner parsers compute the parse probability of a given sentence [e.g., Moore, 2004, Manning and Carpenter, 1997], hence assume a deterministic *shift* move, here we are interested in the joint probability of the parse and the sentence. It will be assumed that the *shift* move requires an additional step in the derivation, connecting an 'incomplete' treelet

(after attach or project) with a word, as illustrated in Figure 6.3. Thus, the derivation is *connected*, and proceeds according to a fixed linear order, which is a prerequisite for the episodic approach. To this end special treelets have been introduced that execute the shift to the next word (e.g., $RC* \rightarrow dances$).[1] These treelets employ special *starred* nonterminals (e.g., $RC*$): one or more stars indicate the register position in the treelet from where the shift operation originates (e.g., $RC \rightarrow WHO \ * \ VI$). The derivation starts with a shift operation from the special $START*$ symbol to the first word of the sentence.

One important difference with the top-down derivation strategy is that upon every attach operation treelets are reengaged in the derivation. It is therefore important to distinguish treelets by their register state, which keeps track of the operations (project, attach) performed on the treelet. Episodic traces are thus associated with and stored in a treelet *in a specific register state*, which is indicated in Figure 6.3 by adding a dot before or after the trace.[2]

6.2.2 Training the episodic grammar

To evaluate the concept of episodic grammar quantitatively a probabilistic version is implemented that is trained on a corpus of realistic language. Probabilistic grammars assign probabilities to different parses of a sentence and select the most probable one, hence can be evaluated on their ability to disambiguate between parses. As explained in section 3.1.5, one estimates the parameters of the probabilistic episodic grammar from a treebank, which is a corpus consisting of natural language sentences manually annotated with phrase structure trees.

After deciding on a derivation strategy (i.e., top-down or left-corner), the training proceeds by distributing a trace $e = \langle s, k \rangle$ in every visited treelet t_k of derivation $x = \langle t_0, \ldots, t_k, \ldots, t_n \rangle$ of sentence number s in the treebank. Specifically, given a treebank, then

1. Create an empty treelet for every unique context free production extracted from the treebank. In case of a left corner derivation one must also create separate treelets for distinct visits to the same production (i.e., after an attach), that is one must distinguish register positions of a treelet. Further, in case of a left corner derivation, create special shift treelets (as described in section 6.2.1) corresponding to the *shift* moves (to terminals) of the left corner parser.
2. For every treebank parse determine the sequential order of (register-indexed) treelets according to the chosen derivation strategy.

[1] This strategy is based on the probabilistic Left Corner Shifting Grammar (LCSG), which will be developed in the next chapter. The LCSG includes shift probabilities, hence defines a language model, which allows for the calculation of sentence probabilities.

[2] In general, there can be as many register positions as there are children in the treelet. In the top-down episodic grammar the register is always in position 0, hence it is not indicated in Figure 6.2.

3. For every step k in the derivation of sentence number s, leave a (register-indexed) trace in the visited treelet, encoded as $\langle s, k \rangle$.

At every derivation step the probability of moving to the next treelet in the derivation can be computed based on the traces in the current treelet and their activations, according to Equation 6.2.

6.2.3 Statistical parsing with the episodic grammar

After training the grammar one can use the model to assign probabilities to candidate parses of a new sentence. Given an ongoing derivation d of a sentence, that has arrived at a certain treelet $t_{q,\,r}$, in register position r, one defines the probability of continuing the derivation to any other treelet $t_{q',\,s}$ in register position s based on the activation values of the episodic traces of earlier derivations stored in treelet $t_{q,\,r}$. The activation $A(e_{x_i})$ of the trace e_{x_i} (in $t_{q,\,r}$) of earlier derivation x is a function of the common history $CH(e_{x_i}, d)$ of derivation x (of which e_{x_i} is the i^{th} trace) with the ongoing derivation d. The CH is simply given by the number of derivation steps (i.e., treelets) that the stored derivation x and the pending derivation d have shared the same path before arriving at $t_{q,\,r}$. Episodic traces that share a long common history should contribute relatively much to the parser decision. A convenient choice for the activation of a trace is

$$A(e_{x_i}) = \lambda_0^{CH(e_{x_i}, d)} \tag{6.1}$$

where λ_0 is a parameter of the model. Depending on the chosen derivation strategy (e.g., top-down or left corner), the traces have different CH's, hence receive different activations.

All information to calculate these activations is stored inside treelet $t_{q,\,r}$; computations are thus local, and compatible with the constraints imposed by a neurally plausible, or connectionist design.

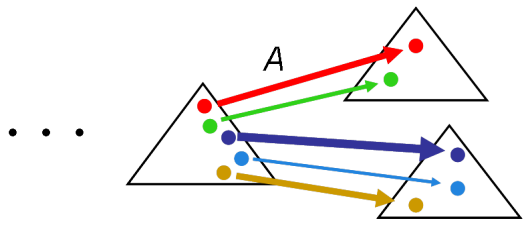


Figure 6.4: Probability of continuing a derivation from treelet t_q to treelet $t_{q'}$ is determined by competition between traces. The width of the arrows indicates the trace activation A.

The probability of moving to $t_{q'}$ in the next step of the derivation is simply the sum of activations of traces that point to $t_{q'}$, divided by the sum of all activations

(see Figure 6.4).[3] Let $E_{t_q}^{t_{q'}}$ be the set of traces in treelet t_q that point to treelet $t_{q'}$, and E_{t_q} the full set of traces in treelet t_q. Then, the probability of moving the derivation to treelet $t_{q'}$ is

$$P_{episodic}(t_{q'}|t_q) = \frac{\sum_{e_i \in E_{t_q}^{t_{q'}}} A(e_i)}{\sum_{e_j \in E_{t_q}} A(e_j)} \tag{6.2}$$

The (episodic) probability of a complete derivation D is given by:

$$P_{episodic}(D = \langle t_0, t_1, \ldots, t_n \rangle) = \prod_{i=1}^{n} P(t_i|t_{i-1}) \tag{6.3}$$

This probability can be computed dynamically, while simultaneously updating the common histories (and activations) of all traces at every step of the derivation. Let t_q and $t_{q'}$ be two successive treelets in the pending derivation d, and let $e' = \langle s, j \rangle$ be a trace stored in $t_{q'}$. Then its CH is updated according to

$$CH(e', d_{q'}) = CH(e, d_q) + 1 \tag{6.4}$$

if there exists a trace $e = \langle s, j-1 \rangle$ in t_q (i.e., a predecessor of e'). Otherwise, $CH(e', d_{q'}) = 0$.

A similar probability model can be derived for the episodic HPN model. There is a complementary role for the *semantic memory* component of HPN (i.e., the metric), namely to provide prior probabilities for transitions between treelets where there is no evidence from previous episodes (i.e., smoothing). One then has

$$P(t_{q',\ s}|t_{q,\ r}) = (1-\lambda) \cdot P_{episodic}(t_{q',\ s}|t_{q,\ r}) + \lambda \cdot P_{semantic}(t_{q',\ s}|t_{q,\ r}) \tag{6.5}$$

For now we focus on the symbolic episodic grammar (with labels), and a full treatment of the episodic-HPN model will be given in Chapter 8.

6.2.4 Smoothing and binarization

In order to obtain a non-zero parse probability for all sentences of the test corpus standard smoothing techniques were performed. Unknown words in the test set were replaced by word classes, which were created from rare words (occurring less than 5 times) in the training set. The word class labels were based on the word's morphology, capitalization, and whether the word occurred at sentence initial position. See [Petrov et al., 2006] for details about the algorithm.

In order to deal with missing productions in the test parse trees, as a first step the rules of the treebank parses were binarized, using horizontal Markovization as

[3] For clarity of notation I have left out the register positions r and s from this point on.

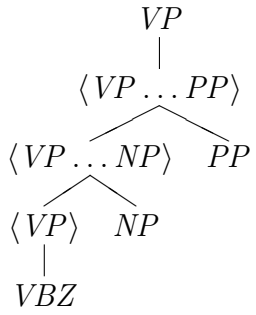


Figure 6.5: Markovization of the tree $VP \rightarrow VBZ\ NP\ PP$ (adapted from [Klein and Manning, 2003])

proposed by [Klein and Manning, 2003]. For any rule with two or more daughters, the right daughters are split off recursively, while the remaining left daughters are replaced with an internal node, as shown in Figure 6.5. In the Figure, the angled brackets (e.g., $< VP \ldots PP >$) indicate internal labels, and the dots summarize all internal labels that expand to VP as their leftmost daughter and PP as their rightmost daughter.

Subsequently, three levels of back-off smoothing (i.e., deleted interpolation) were used, where every level conditioned on less context (see Equation 6.6). The first level back-off probabilities, P_1, backs off to a non-episodic version of the chosen derivation strategy. In the top-down episodic grammar these are the PCFG rule probabilities, which condition the application of a treelet on a single, expanding nonterminal label; In the case of a left corner episodic grammar the first level backs off to a standard probabilistic left corner model. This conditions the application of a treelet on the left corner and goal category, following [Manning and Carpenter, 1997].

The second level, P_2, backs off the conditioning context of any compound nonterminal (originating from the Markovization step) by reducing the conditioning context of a label to its left element alone (e.g. X in $< X \ldots Y >$). Thus, given a unary or binary PCFG rule with a compound root label, e.g., $< X \ldots Z > \rightarrow < X \ldots Y >\ Z$, the backed off probabilities $P(< X \ldots Y >\ Z|X)$ generalize over all such rules with arbitrary Z that have X as the left element of their root nonterminal. Similarly, in the left corner grammar the second level backs off a compound left corner label to its left-most element.

The third level, P_3, assigns uniform probabilities to all possible unary and binary context free productions (that can be constructed from the nonterminals of the grammar), irrespective of context. The three levels are parametrized by back-off parameters λ_1, λ_2 and λ_3, yielding

$$P(t_{q'}|t_q) = (1-\lambda_1) \cdot P_{episodic} + \lambda_1 \cdot ((1-\lambda_2) \cdot P_1 + \lambda_2 \cdot ((1-\lambda_3) \cdot P_2 + \lambda_3 \cdot P_3)) \quad (6.6)$$

In this equation the λ's are fixed, and all back-off probabilities are estimated from the training corpus.

6.2.5 Evaluation and reranking

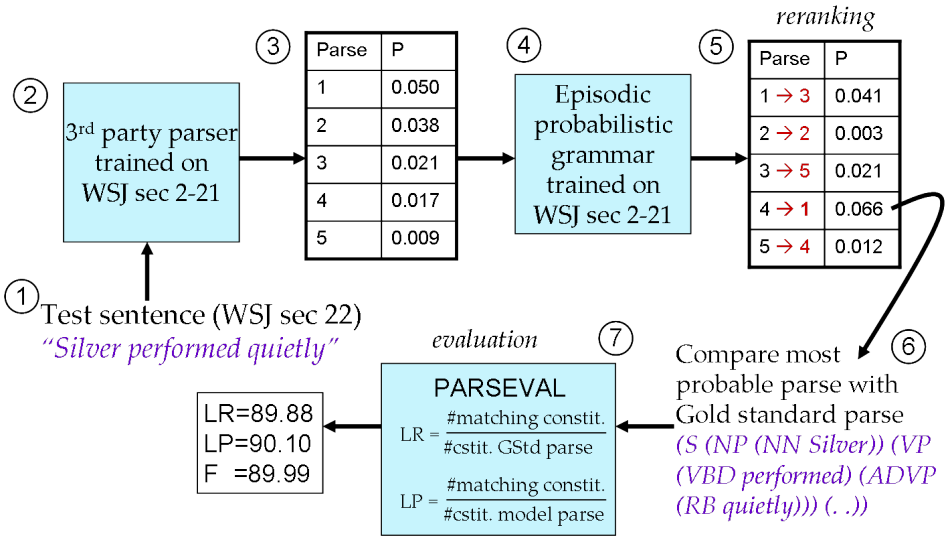


Figure 6.6: The reranking process.

As has become the standard, the episodic grammar was trained on sections 2-21 from the Penn Wall Street Journal corpus (WSJ) [Marcus et al., 1993] and evaluated on section 22 of WSJ. For the test section labeled precision and recall of the most probable parses according to the model were measured using the PARSEVAL metric (see section 3.1.6).

While in Chapter 7 I will develop a specialized left corner chart parser for the episodic grammar, at this stage it is interesting to study the properties of the episodic probability model, and a straight forward way to do this is to use the model as a *reranker*. This means that one takes a list of n best parses for every sentence produced by a third party parser (in this case Charniak's maximum entropy parser [Charniak, 2000]), and reranks the list by assigning a probability to each parse under the model of interest [Sangati et al., 2009]. One can then use the standardized PARSEVAL metric to evaluate labeled precision (LP), labeled recall (LR) and their harmonic mean (F-score) of the parses that receive the highest probability under the reranker [Manning and Schütze, 2000, p. 432]. Figure 6.6 illustrates the reranking process step by step.

Reranking does have some limitations as an assessment of the model's performance, since the n best parses list produced by the third party parser has upper and lower bound precision and recall scores. For comparison the scores are given of a random reranker, that selects a parse from the list by chance. Confidence in

the results of the reranker increases with the size n of the list of the best third party parses (NBest list) (e.g., see Figure 6.8).

6.3 Experiments and results

The precision and recall results of the episodic top-down reranker, applied to the top 5 Charniak parses, are given in the first three columns of Table 6.1 as a function of the maximum common history that is taken into account by the episodic grammar (the column *max his*). CH's larger than the maximum history are capped in equation 6.1. The bottom 2 rows give the Charniak scores and the scores for a random reranker; As is common practice, only sentences of 40 words or less were included. I have experimented with different parameterizations of $\lambda_0, \ldots, \lambda_3$ on the development set . Optimal results were obtained for λ_0=4, and $\lambda_1, \ldots, \lambda_3$ in the range between 0.1-0.3, with only little variance. In Table 6.1 and

	top down reranker			left corner reranker		
max his	LR	LP	F	LR	LP	F
0	87,11	90,01	88,54	87,93	90,31	89,10
1	89,53	90,27	89,90	89,35	90,22	89,79
2	89,64	90,23	89,94	89,49	90,30	89,89
3	90,15	90,45	90,30	89,64	90,43	90,04
4	90,15	90,39	90,27	89,79	90,53	90,16
5	**90,27**	**90,45**	**90,36**	89,91	90,63	90,27
6	90,23	90,41	90,32	89,96	90,58	90,27
7	90,19	90,37	90,28	90,13	90,76	90,44
8	90,09	90,21	90,15	**90,32**	**90,90**	**90,61**
9	90,14	90,27	90,20	90,29	90,84	90,56
10	90,03	90,16	90,09	90,23	90,79	90,51
11	89,98	90,14	90,06	90,10	90,74	90,42
12	89,91	90,11	90,01	90,07	90,67	90,37
Ch	90,23	90,15	90,19	90,23	90,15	90,19
Ran	88,15	87,89	88,02	88,17	87,84	88,00

Table 6.1: Precision and recall scores of the episodic *top-down* reranker (columns 1-3) and *left corner* reranker (columns 4-6) as a function of the maximum history considered ($nBest$=5; λ_0=4; λ_1=λ_2=λ_3=0.2).

Figure 6.7 one can see a clear effect of conditioning history, peaking at history 5 for the top-down reranker, and at history 8 for the left corner reranker (best scores are indicated in boldface). For histories 3-7 the episodic top-down reranker surpasses the Charniak F-scores by a slight margin, and overall does much better than the PCFG reranker (corresponding to *history* 0) and the random reranker.

As can be seen from Table 6.1, the LCE grammar performs better across the board than the TDE grammar, and this is mainly due to improved labeled precision scores. It also does better than the probabilistic left corner model of [Manning and Carpenter, 1997], which corresponds to the top row in the Table. Note that for the LCE reranker the peak is reached at history 8, and the F-scores

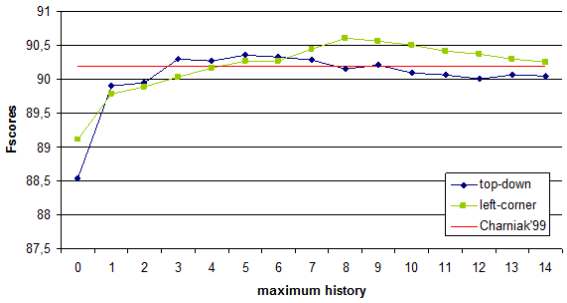


Figure 6.7: F-scores compared between the top-down and the left corner episodic reranker as a function of conditioning history.

stay high until history 14; this could be an indication that the order of conditioning in a LCE derivation better approximates human sentence processing than in a TDE derivation. It is remarkable that the LCE grammar robustly improves on the Charniak parser, because i) unlike the latter it does not implement head annotation or other non-trivial preprocessing steps, ii) it makes several non-standard assumptions about the derivation process, such as a left-corner sequential order and the inclusion of special shift treelets in the derivation for transitions from incomplete productions to words.

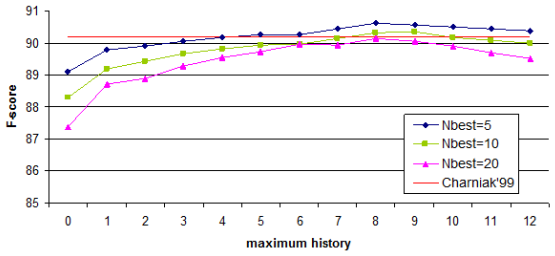


Figure 6.8: F-scores of the left corner episodic reranker applied to the top 5, top 10 and top 20 Charniak parses.

To assess the robustness of the reranking method I have also applied the LCE reranker to the top 10 and the top 20 lists of Charniak parses. In the latter case the random reranker baseline is significantly lower than for the top 5 reranker (F-score = 86.2 resp. 88.0). Therefore it is meaningful that the top 20 reranker still performs almost as good as Charniak (F-score=90.15 for history 8), and the top 10 reranker does even better (F-score=90.34 for history 9). In Figure 6.8 it can further be seen that although the differences in performance between the top 5, top 10 and top 20 reranker are large for low histories, they converge for histories of 6-10, when the episodic approach starts to make a difference. On the other hand, the TDE reranker breaks down when applied to the top 20 Charniak parses, peaking at an F-score of 89.66 for history 6.

6.3.1 Discontiguous episodes

An interesting way to extend the episodic grammar is by including *discontiguous* episodes. Often one can reuse a memorized sentence fragment, even if it does not exactly match the sentence that is currently being processed, but differs from it by a single word or clause. I implemented a variation of update rule for the common history (CH) in order to include episodes with 'gaps'. In Equation 6.4, whenever an episode is interrupted (i.e., its CH is set to 0) it is pushed together with its current activation on an external stack of discontiguous episodes (a separate stack is used for every exemplar). If at a later stage in the derivation a trace of the same exemplar is found, which has no predecessor, then one can pop up an interrupted episode from the top of the stack of that exemplar, and copy (a fixed fraction f of) its activation to the new trace.

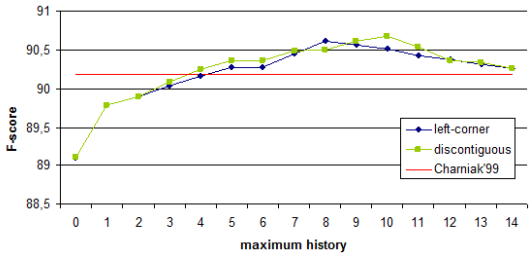


Figure 6.9: F-scores of the LCE reranker with and without counting discontiguities (d=0.95; f=0.6)

.

Best results were obtained when the activation of unused discontiguous episodes decays by some percentage d at every step of the derivation. With $d = 0.95$ and $f = 0.6$ the addition of discontiguous episodes gives a minor improvement over the non-discontiguous case, as can be seen from Figure 6.9. The highest F-score is 90.68, which is reached for history 10. The effect of the inclusion of discontiguous fragments seems to be that longer histories play a more prominent role.

If one looks at the individual sentences from the test set (WSJ section 22) for which the F-scores increased most by including discontiguous fragments, one finds that those are indeed sentences that employ frequent discontiguous expressions. For instance, within the top 5 of these sentences one contains the discontiguous fragment *rose to ... from ...*, which occurs more than 100 times in the training corpus.

6.3.2 Shortest derivation reranker

Assuming that language users understand and produce novel sentences by reusing fragments of stored episodes, then intuitively they will try to do so by retrieving not only the most frequent, but also as few as possible fragments from memory,

since this demands the least cognitive effort. This amounts to a preference for the shortest derivation of a novel sentence.

Such a preference can be implemented in the episodic grammar framework by greedily selecting fragments from stored exemplars that share the largest common history with the derivation of a novel sentence (not including fragments from exemplars that are identical to the novel sentence). When the shortest derivation principle is used together with the LCE reranker to select those derivations of the Nbest list that use the fewest episodes (followed by selection of the derivation with the highest likelihood in case of a draw) then an F-score of 90.44 is obtained (for history 9). Thus, the shortest derivation LCE reranker performs worse than the maximum likelihood LCE reranker, but still better than the Charniak parser.

In Data Oriented Parsing the principle of the shortest derivation has been successfully explored as an alternative to a probabilistic parsing strategy [Bod, 2000]. The multi-word fragments employed in the shortest derivations (or in the most probable derivations) are assumed to have some cognitive reality as the primitive building blocks of speech. In the DOP framework however a top-down derivation is always assumed, whereas in the episodic framework one can also find fragments of a left corner derivation. Figure 6.10 shows some examples of frequent fragments that occur in the shortest derivations of the Tuebingen Corpus of English Spontaneous Speech (www.sfs.uni-tuebingen.de/en/tuebaes.shtml), when the parse trees are derived with a left corner episodic grammar.

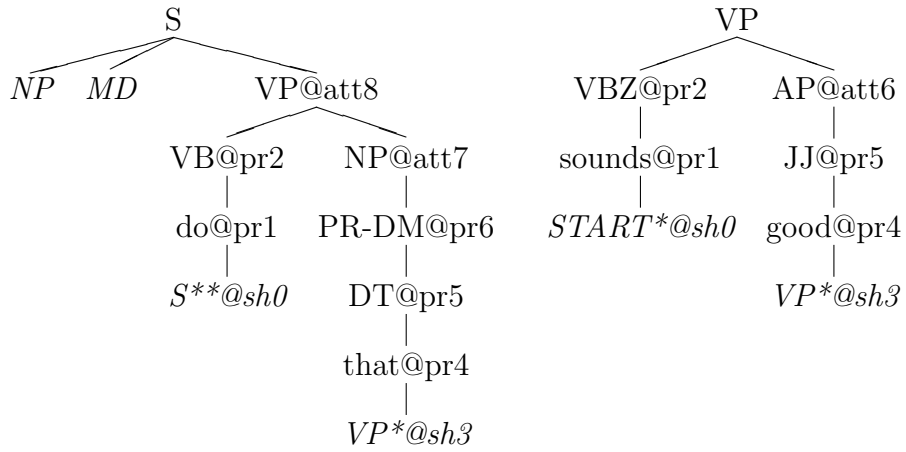


Figure 6.10: Examples of frequent fragments used in the shortest derivations of the Tuebingen corpus. The letters after the @-symbol indicate the applied operation (sh(ift), att(ach), pr(oject)), and the order of application.

6.4 Relation to other work

As was discussed in section 3.1.8, current research in statistical NLP and parsing increasingly focuses on ways to weaken the context independence assumptions of probabilistic context free grammars (PCFGs). Context free grammars fail to take advantage of two relatively independent sources of contextual information for disambiguating between parses: *context!structural and lexical*, which captures the dependency on previous words in the sentence, and *structural context*, which captures the dependency on the relative position in a parse tree. In section 3.1.8 I have discussed some of the solutions that have been investigated, such as *head lexicalization* and *parent annotation*; all of these involve transferring contextual information to the labels of the trees as to preserve the context free backbone of the grammar.

In the episodic grammar both lexical context and structural context are integrated in the conditioning history without any need for preprocessing of the labels. For instance, in the LCE grammar all words to the left of the currently processed word weigh in the parser move decision. As such, the LCE grammar should be considered as a good candidate for language processing.

Parsing with episodic grammars is in some respects comparable to the tradition of *history based parsing*, which exploits the idea that the parser moves are conditioned on n previous parser decisions in the derivation history. A weakness of the latter approach is however that it leads to very large grammars and data sparsity, since all conditioning events are saved explicitly in equivalence classes [e.g., Black et al., 1993, Collins, 1999, p.57]. In the episodic grammar parser decisions are conditioned on arbitrary long histories, at no cost to the grammar size, because conditioning context is implicit in the representation, and is constructed explicitly only during on-line processing of a novel sentence. Since every exemplar is stored only once in the network, the space complexity of the episodic grammar is linear in the number of exemplars.

Another difference with history-based parsers is that in the latter the association between the conditioning event and the sentence from which it originates is lost, whereas in the episodic grammar the identity of an exemplar that has contributed to a derivation step is preserved. In section 6.3.1 it was shown that this feature can be used for including discontiguous episodes.

It is also interesting to compare the episodic grammar with Data Oriented Parsing (DOP) [e.g., Bod, 1998] (see section 3.1.9). In DOP the primitive units of the grammar are not CF rules, but subtrees of arbitrary size, which are extracted from the parses of a treebank. In a certain sense DOP and episodic parsing are complementary: whereas in DOP the substitution of an arbitrary large subtree is conditioned on a single nonterminal, in the episodic parser the application of a local tree is conditioned on an arbitrary large episode. However, the shortest derivation variant of the episodic reranker effectively combines both conditioning on large histories and substitution of stored units larger than a single treelet.

Further, both approaches allow for non-local dependencies to be captured in primitive, discontiguous fragments of the grammar, but in the episodic framework this is less straight forward to implement than in DOP. An advantage of episodic grammar over DOP is that in the former the stored parse tree can be broken down into subtrees according to various generative processes (top-down, left corner, or any other decomposition) whereas in DOP always a top-down generative process is assumed. This opens the possibility to utilize the episodic grammar as a language model in speech recognition, for which a left corner strategy is more suitable than a top-down strategy.

As was mentioned before, in the episodic grammar it is not necessary to store every possible tree fragment explicitly. This is an advantage over DOP, which suffers from computational inefficiency due to very large grammars. The fact that stored episodes are automatically *reconstructed* from traces during the derivation of a novel sentence obviates a time-expensive search through an external memory (i.e., a treebank of fragments), and makes the episodic grammar *content-addressable*.

Table 6.2 shows how the present results compare to state-of-the-art parsers. Note that the latter are evaluated on section 23 of WSJ, while all the results of this work are on section 22. Note also that for the present results a reranker is used, that is parasitic on the Charniak (1999) parser.

Various parser strategies (on WSJ sec 23)

Parsing model	F ($\leq$ 40)	F (all)
Charniak (1999) (max. entropy)	90.1	89.6
Petrov and Klein (2007) (refinement-based)	90.6	90.1
Bansal and Klein (2010) (fragment-based)	88.7	88.1
Sangati & Zuidema (2011) (DOP)	89.7	89.1
Cohn et al. (2009) (Bayesian)	-	84.0
Charniak and Johnson (2005) (reranker, $n = 50$)	-	90.1
This paper (on WSJ sec 22)		
TDE reranker ($n = 5$)	90.4	-
LCE reranker ($n = 5$)	90.6	90.1
LCE + disctg ($n = 5$)	90.7	-

Table 6.2: Comparison of the episodic reranker to state-of-the-art parsers, for sentences of length up to 40, or all sentences.

6.5 Chapter conclusion

In this chapter I described a cognitively inspired implementation for contextual conditioning in statistical parsing, using episodic memory. It was shown that for the task of supervised parsing the episodic grammar is a viable alternative for standard, not cognitively motivated probabilistic grammars. At the same time the episodic grammar offers a neural perspective on human syntax, that unifies

the contrasting views that syntax is either encoded as a set of abstract rules, or as stored exemplars of (fragments of) sentences.

It will be even more interesting to see whether the episodic framework can be successful as an approach to the *unsupervised* induction of (neurally plausible) grammars from unannotated sentences. Since in episodic parsing all computations are done locally, the framework is in principle compatible with the constraints imposed by a connectionist design. This will be explored in Chapter 8, where I will evaluate an episodic version of HPN.

The current work should not only be seen as an exercise in computational linguistics, but also as a theoretical contribution to episodic memory research. As such, it is an instance of how cognitively inspired linguistic research can open a window on the study of memory processes in the brain. I proposed an original hypothesis for the representation of episodic memory, which expresses that an episodic memory is distributed in the form of traces, supplied with a time stamp, *inside* local stores of the semantic memory units that are involved in processing it. According to a free interpretation of this proposal one could imagine episodic memory as a life-long thread spun through semantic memory.

In contemporary theoretical neuroscience most models of episodic memory assume dedicated 'binding neurons', whose sole job it is to bind semantic 'content' nodes into episodic representations [e.g., MacKay, 2007, Shastri, 2002, O'Reilly and Rudy, 2001, O'Reilly and Norman, 2002]. Yet, this is not a very feasible solution for the representation of episodic memories, for every day of a person's life many thousands of new episodic memories are formed. If episodic memories were stored in binding units, this would require the neurogenesis of a massive number of neurons and the establishment of even more new connections. As will be explained in section 8.4.2, in the current proposal successive traces of an episode are assumed to be dynamically bound, hence binding neurons are not necessary. In this sense the current proposal, although simple, contributes to the episodic memory debate, because it shows a way out of the curse of connectivity.

In their essence, the ideas developed in this chapter are consistent with contemporary research in neuroscience, which emphasizes the construal of episodes in the hippocampus as contextually bound sequences of semantic memories [e.g., Eichenbaum, 2004] (see section 8.4 for a discussion of the episodic-HPN model in the neuro-biological context). The hippocampal model of Levy [1996] shows that during episodic sequence learning special 'context neurons' are formed that uniquely identify (part of) an episode. These may function as a neural correlate of the *counter* that was implemented in the traces. The episodic grammar model represents a first attempt to validate this theory of episodic memory within the language domain.

Chapter 7

Parsing with episodic memory

While the previous chapter described the episodic probability model and used it as a reranking system, in this chapter I will take the episodic framework one step further and develop a full episodic left corner chart parser. As a first step a non-episodic probabilistic left corner chart parser that computes prefix probabilities is introduced in reasonable detail. Subsequently I describe an episodic 'spreading activation' instantiation of the left corner chart parser (ELCCP) where episodic probabilities are computed on-the-fly at parse time through spreading trace activations from stored derivations to (the traces in) its states. I discuss an efficient Viterbi algorithm for dynamically computing the shortest derivation within the ELCCP, which is implemented and evaluated on the Wall Street Journal.

7.1 An Earley-style probabilistic left-corner chart parser that computes prefix probabilities

In this section and the following I will gradually build an episodic left corner chart parser for HPN. Recall from section 5.3.1 that top-down parsing is not an option for HPN, because no priviliged TOP category nor any other labels are assumed, and there are no restrictions on binding between HPN units, hence the search could go on indefinitely. As a first step, this section describes an Earley-style probabilistic left-corner chart parser that computes prefix probabilities. The left corner chart parser described here follows in broad lines the work of van Uytsel

et al. [2001] (which unfortunately I learned about only after I had developed a very similar left corner chart parser myself), and it is based on the Earley chart parser [Earley, 1970] (see section 3.1.11) and the probabilistic version thereof [Stolcke, 1995] (see section 7.1.6).

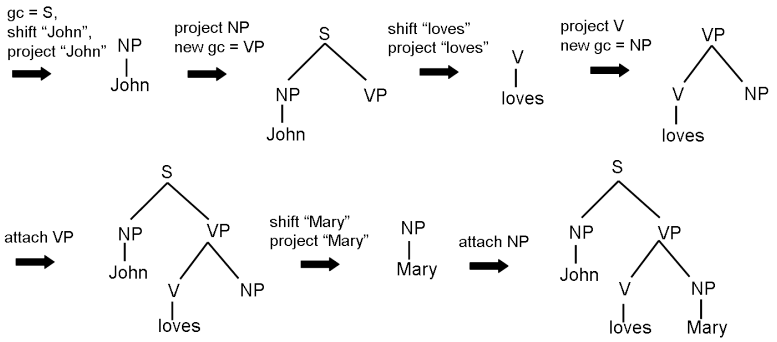


Figure 7.1: Left corner derivation of *John loves Mary.* gc is short for goal category.

Whereas in Stolcke [1995] a top-down parsing process is assumed (and a PCFG probabilistic model) the chart parser of van Uytsel et al. [2001] follows a left-corner parsing (LCP) strategy (see section 3.1.3), and a left corner probabilistic model, which will in general return different string probabilities than the top-down model. As discussed in section 3.1.3 the left corner parser has three operations: *shift*, *attach* and *project*. As a reminder, Figure 7.1 (repeated from section 3.1.3) illustrates the left corner parsing process.

7.1.1 States of the left corner parser

The LC parsing process is executed as a search through a network of *states* in a chart, where state transitions are effectuated by the *shift*, *project* and *attach* operations. States represent an instantiation of a grammar rule within a derivation: they are distinguished with respect to their position in the input string and their goal category, and are of the form

$$q = \{G;\ X \leftarrow {}_j\lambda\ \bullet {}_i\mu\} \tag{7.1}$$

where G is a goal category, $X \leftarrow \lambda\ \mu$ is a grammar rule[1], and λ and μ are (possibly empty) strings of terminals and nonterminals. The position of the dot indicates which daughters of the rule have already been processed in the state. If the dot is on the right of all daughters of a rule the state is called *complete*; otherwise, if it is not complete, it is called a *goal state*. The *left span index* j of state q

[1]To prevent confusion I have adopted a notation where the arrow points left, instead of right, as customary in context free rewrite rules. The right arrow would be suggestive of a top-down prediction, whereas in left corner parsing the rule is accessed first through bottom-up projection.

($lspan(q)$, for short) corresponds to the word position in the sentence when the rule was first *projected* (i.e., words $x_0, \ldots, x_{j-1}$ have been processed before the projection); the *right span index* i ($rspan(q)$, for short) denotes the word position of the dot with respect to the input string. In plain language, $X \leftarrow {}_j\lambda \bullet_i \mu$ means that the processed part of the rule spans $< j,\ i >$ in the input string.

Most probabilistic left corner parsers [e.g., Manning and Carpenter, 1997] compute the parse probabilities *given* the sentence (see section 3.1.7). If one is however interested in sentence or string probabilities (e.g., for a language model, or to compute prefix probabilities) this introduces some additional challenges, because a left corner derivation may produce many *unconnected* partial substructures (section 3.1.3). In order to compute the probability of a certain prefix one needs to make assumptions about how the substructures relate to each other in the generation process of the tree.

To deal with this problem one may postulate that in a LC derivation a goal state is followed by a special *word state* (a state of the parser after a shift), and that this transition is mediated by a *shift rule* from the *goal category* (the first unprocessed daughter of the goal state) to the word (for instance *loves* $\leftarrow$ *VP* in Figure 7.1). With this assumption a LC derivation always stays connected, since it defines a unique and fixed linear order of processing. It is convenient to write the shift rule as

$$word \leftarrow G$$

where G is the goal category; the word state is then given as

$$\{G;\ word \leftarrow {}_iG \bullet\ {}_{i+1}\}$$

The left corner grammar augmented with a shift rule will be called *Left Corner Shifting Grammar* (LCSG).

7.1.2 Probabilistic left corner shifting grammar

A left corner shifting grammar can be extended to a probabilistic LCSG (PLCSG) by assigning a probability to every operation. Whereas in a PCFG there is only one type of probability, associated with top-down prediction and conditioned on the parent in the tree, in the PLCSG there are three types of probabilities, corresponding to different parser moves (project, attach and shift). The left corner probabilities are conditioned on previous steps in the derivation (history-based parsing), rather than based on the tree structure.

Often one or two prominent features are selected from the history to condition upon. For instance, Manning and Carpenter [1997] condition the probability of a projection and of an attachment on the parent category (Y) of a completed rule and the goal category (G) in the stack of the parser. Their example will be followed in the LCSG parser, but in a later section I will show that an *episodic* LCSG parser is able to condition on the entire history of the derivation.

As discussed in section 3.1.7, given a *complete* state with parent category Y, and given a goal category G, one can either project a rule $r: \ Z \leftarrow Y \ \alpha$ with left corner Y, or attach to a goal state if its first unprocessed daughter G equals Y, hence

$$P_{att}(Y, G) + \sum_{Z, \ \alpha} P_{proj}(Z \leftarrow Y \ \alpha | Y, G) = 1 \tag{7.2}$$

If there are no complete states then one must shift from a goal state to the next word in the sentence, conditioned on the goal category G (the first unprocessed daughter of the goal state). The *shift probability* is therefore given as

$$\sum_{word} P_{shift}(word|G) = 1 \tag{7.3}$$

The above probabilities can be estimated from the treebank using relative frequency estimation, after converting the treebank parses to their left corner derivations. Given shift probabilities one can calculate the joint probability of the derivation and the sentence, which is given by

$$\begin{array}{rcl} P(der_{lc}, S) = \prod_{shifts} P_{shift}(w|G) & \times & \prod_{attachments} P_{att}(Y, G) \\ & \times & \prod_{projections} (1 - P_{att}(Y, G)) \times P_{proj}(r|Y, G) \end{array} \tag{7.4}$$

7.1.3 Probabilistic left corner chart parsing

As one can define a dynamic programming chart parser for the probabilistic top-down Earley-style chart parser of [Stolcke, 1995], one can do so as well for the LCSG grammar, in a way that allows for computing prefix probabilities and find the most probable parse efficiently. In section 7.1.6 I gave definitions for the prefix probability, the forward probability (for short, P_{fw}) and the inner probability (for short, P_{inn}) for the top-down parser. These definitions can be accommodated for the left corner parser, taking into account the different branching process of the left corner search. Given a state q as defined in Equation 7.1, then one defines

Forward probability The forward probability $P_{fw}(q)$ is the sum of the probabilities of all constrained paths of length j that end in state q, start in the initial state and generate $x_0 \ldots x_{j-1}$.

Inner probability The inner probability $P_{inn}(q)$ is the sum of the probabilities of all paths of length $k - j$ generating the input symbols $x_j \ldots x_{k-1}$, ending in q and starting with a *shift* of w_j.

(The prefix probabilities can be easily computed from the forward probabilities, as will be discussed in section 7.1.4.) One may also make use of the LCSG chart structure to efficiently find the most probable parse of the sentence (a.k.a.

the *Viterbi parse*). We define the Viterbi probability of a state, $P_{Vit}(q)$, as the probability of the most probable path (the *Viterbi path*), constraint by the sentence string, that goes through state q. To find the Viterbi path, the states in the chart must update their Viterbi probabilities, and keep track of a *Viterbi pointer* to the predecessor states associated with the Viterbi path. After completing the chart one can reconstruct the Viterbi parse from the final state by tracing back the pointers.

The LCSG chart parser moves through the sentence from left to right, and it keeps a set of states for each position in the input. Starting from state set 0, and as long as there are complete states in the current state set, the parser adds new states to the chart by exhaustively, and recursively performing *project* and *attach* operations on complete states.[2] Then it adds states to the next state set by *shifting* to the next input symbol from all goal states in the current state set.

A derivation starts with the initial state

$$q_I = \{TOP;\ TOP \leftarrow\ _{-1}\ SB \bullet\ _0\ S\}$$

which is placed in state set 0 (SB is a dummy nonterminal that marks the Sentence Beginning).[3] Since this is a goal state, the first operation will be a shift operation, invoking a shift rule $x_0 \leftarrow TOP$ (with x_0 the first word of the sentence). The parse completes when it reaches the final state

$$q_F = \{TOP;\ TOP \leftarrow\ _{-1}\ SB\ S \bullet\ _N\}$$

where N equals the number of words in the sentence.

Following are the details of the LCSG chart parser operations, specifying the conditions for adding new states to the chart, and the dynamic update rules for the forward, inner and Viterbi probabilities.

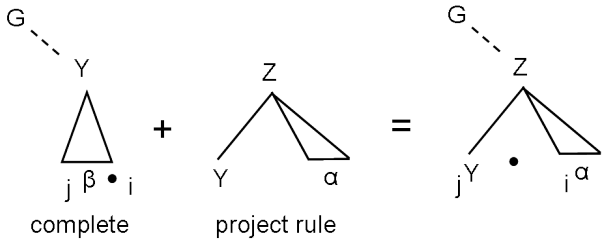


Figure 7.2: Project operation

- The *project* operation (Figure 7.2)
  Given a complete state $\mathcal{C} = \{G;\ Y \leftarrow\ _j\beta\ \bullet\ _i\}$ with parent category Y

[2]There are some intricacies involved here, concerning keeping the correct order and generating states recursively. These will be dealt with in section 7.1.6.

[3]This assumes that all parses must have a root category S.

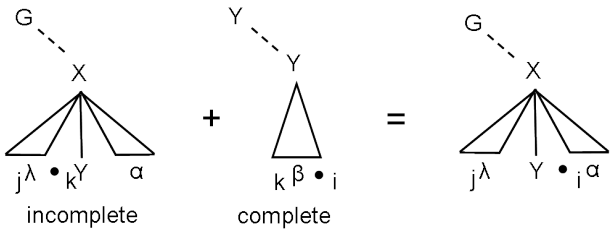


Figure 7.3: Attach operation

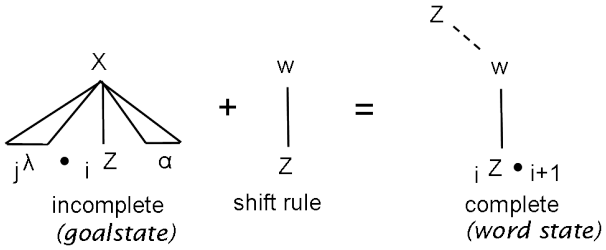


Figure 7.4: Shift operation

and a grammar rule $Z \leftarrow Y\ \alpha$ with left corner Y
If it does not yet exist, then create a new projected state
$\mathcal{N} = \{G;\ Z \leftarrow {}_jY \bullet {}_i\ \alpha\}$ with the dot in the second position, and add it to the chart.

The forward, inner and Viterbi probability are updated according to:

$$P_{inn}(\mathcal{N}) \mathrel{+}= P_{inn}(\mathcal{C}) \times P_{proj}(r|G) \tag{7.5}$$

$$P_{fw}(\mathcal{N}) \mathrel{+}= P_{fw}(\mathcal{C}) \times P_{proj}(r|G) \tag{7.6}$$

$$P_{Vit}(\mathcal{N}) = \max\left(\left(P_{Vit}\left(\mathcal{C}\right) \times P_{proj}\left(r|G\right)\right), P_{Vit}^{old}\left(\mathcal{N}\right)\right) \tag{7.7}$$

The notation $+=$ means that the probabilities are set to the value on the right hand side of the equation if $\mathcal{N}$ does not yet exist in the chart; otherwise their values are incremented with the same amount. If $\mathcal{N}$ already exists in the chart, the Viterbi probability is only updated if it is higher than the state's previous Viterbi probability $P_{Vit}^{old}(\mathcal{N})$. In that case the Viterbi pointer is also replaced.

- The *attach* operation (Figure 7.3):
  Given a complete state $\mathcal{C} = \{Y;\ Y \leftarrow {}_k\beta \bullet {}_i\}$ and an incomplete goal state $\mathcal{I} = \{G;\ X \leftarrow {}_j\lambda \bullet {}_k\ Y\ \alpha\}$, where $rspan(\mathcal{I}) = lspan(\mathcal{C}) = k$.
  If it does not yet exist in the chart, create a new state
  $\mathcal{N} = \{G;\ X \leftarrow {}_j\lambda\ Y \bullet {}_i\alpha\}$, in which the dot is moved behind Y.

Let $P_{att}(Y, G)$ be the probability of attaching a complete state with parent category Y to goal category $G = Y$. Then the inner, forward and Viterbi probability are updated according to

$$P_{inn}(\mathcal{N}) \mathrel{+}= P_{inn}(\mathcal{C}) \times P_{inn}(\mathcal{I}) \times P_{att}(Y, G) \tag{7.8}$$

$$P_{fw}(\mathcal{N}) \mathrel{+}= P_{inn}(\mathcal{C}) \times P_{fw}(\mathcal{I}) \times P_{att}(Y, G) \tag{7.9}$$

$$P_{Vit}(\mathcal{N}) = \max\left((P_{Vit}(\mathcal{C}) \times P_{Vit}(\mathcal{I}) \times P_{att}(Y, G)), P_{Vit}^{old}(\mathcal{N})\right) \tag{7.10}$$

The new state can be either complete (if $\alpha = \varnothing$) or incomplete (if $\alpha \neq \varnothing$). The rationale for updating the forward probabilities from the inner probabilities will be explained with an example in section 7.1.5.

- The *shift* operation (Figure 7.4):
  Given an *incomplete* goalstate $\mathcal{I} = \{G;\ Y \leftarrow {}_j\lambda \bullet {}_iZ\ \alpha\}$, and given a shift rule $w \leftarrow G$
  If it does not yet exist in the chart, create the complete state
  $\mathcal{N} = \{Z;\ w \leftarrow {}_iZ \bullet {}_{i+1}\}$

$$P_{inn}(\mathcal{N}) = P_{shift}(w|G) \tag{7.11}$$

$$P_{fw}(\mathcal{N}) \mathrel{+}= P_{fw}(\mathcal{I}) \times P_{shift}(w|G) \tag{7.12}$$

$$P_{Vit}(\mathcal{N}) = P_{shift}(w|G) \tag{7.13}$$

To comply with [Stolcke, 1995] I will call a complete state that results from a shift a *scanned state*.

7.1.4 Prefix probabilities

Recall from section 7.1.6 that the prefix probability $P(S \leftarrow_L^* x)$ is the sum of the probabilities of all derivational paths starting with the initial state, that have $x = x_0, \ldots, x_{k-1}$ as a prefix. Since all derivations of sentences with prefix $x_0, \ldots, x_{k-1}$ (and $|x| = k$) have to go through a scanned state $\{G;\ x_{k-1} \leftarrow {}_{k-1}\ G\ \bullet\ {}_k\}$, one may compute the prefix probability as the sum of the forward probabilities of all such scanned states. Hence,

$$P(S \leftarrow_L^* x) = \sum_{G:\ \{G;\ x_{k-1} \leftarrow_{k-1}\ G\ \bullet\ {}_k\}} P_{fw}(\{G;\ x_{k-1} \leftarrow {}_{k-1}\ G\ \bullet\ {}_k\}) \tag{7.14}$$

Figure 7.5: Treebank parses of *Peter runs* (PRN = pronoun).

7.1.5 An example that explains why inner probabilities are necessary

Suppose the left corner parser of van Uytsel et al. [2001] is trained on a treebank containing the two parses of Figure 7.5. Table 7.1 gives the states that the left corner chart parser goes through when parsing the sentence *Peter runs*.

nr	*op.*	*from*	*state*	*p*	P_{fw} (μ)	P_{inn} (ν)
q_I	-	-	$TOP;\ TOP \leftarrow_{-1} SB \bullet_0 S$	-	1.0	1.0
q_2	*sh*	q_I	$S;\ Peter \leftarrow_0 S \bullet_1$	$P_{sh}(Peter\|S) = 1.0$	1.0	1.0
q_3	*pr*	q_2	$S;\ NP \leftarrow_0 Peter \bullet_1$	$P_{pr}(NP\|Peter, S) = 0.6$	0.6	0.6
q_4	*pr*	q_2	$S;\ PRN \leftarrow_0 Peter \bullet_1$	$P_{pr}(PRN\|Peter, S) = 0.4$	0.4	0.4
q_5	*pr*	q_3	$S;\ S \leftarrow_0 NP \bullet_1 VP$	$P_{pr}(S, VP\|NP, S) = 1.0$	0.6	0.6
q_6	*pr*	q_4	$S;\ S \leftarrow_0 PRN \bullet_1 VP$	$P_{pr}(S, VP\|PRN, S) = 1.0$	0.4	0.4
q_7	*sh*	$q_5,\ q_6$	$VP;\ runs \leftarrow_1 VP \bullet_2$	$P_{sh}(runs\|VP) = 1.0$	1.0 ($\mu_5 + \mu_6$) · P_{sh}	1.0
q_8	*pr*	q_7	$VP;\ VP \leftarrow_1 runs \bullet_2$	$P_{pr}(VP\|runs, VP) = 1.0$	1.0	1.0
q_9	*att*	$q_5 + q_8$	$S;\ S \leftarrow_0 NP\ VP \bullet_2$	$P_{att}(VP, VP) = 1.0$	0.6 ($\mu_5 \cdot \nu_8 \cdot P_{att}$)	0.6 ($\nu_5 \cdot \nu_8 \cdot P_{att}$)
q_{10}	*att*	$q_6 + q_8$	$S;\ S \leftarrow_0 PRN\ VP \bullet_2$	$P_{att}(VP, VP) = 1.0$	0.4 ($\mu_6 \cdot \nu_8 \cdot P_{att}$)	0.4 ($\nu_6 \cdot \nu_8 \cdot P_{att}$)
q_F	*att*	$q_9 + q_I$, $q_{10} + q_I$	$TOP;\ TOP \leftarrow_{-1} SB\ S \bullet_2$	$P_{att}(S, S) = 1.0$	1.0 ($\mu_1 \cdot \nu_9 \cdot P_{att} + \mu_1 \cdot \nu_{10} \cdot P_{att}$)	1.0 ($\nu_1 \cdot \nu_9 \cdot P_{att} + \nu_1 \cdot \nu_{10} \cdot P_{att}$)

Table 7.1: Left corner chart of the sentence *Peter runs*. (Note that in the Table P_{fw} is abbreviated as μ and P_{inn} as ν.)

Why can one not compute the forward probabilities efficiently without inner probabilities? This has to do with the existence of non-local dependencies in the left corner derivations. One can see from the chart and from Figure 7.6, that after a shift to *runs*, both the path through state q_5 and the path through q_6 converge on state q_7. Therefore, q_7 receives a contribution to its forward probability from both μ_5 and μ_6. However, when the same paths attach back to q_5 and q_6 respectively (in q_9 and q_{10} respectively) they are again separated. Therefore, one cannot use the (summed) forward probability of q_7 in the calculation of both μ_9 and μ_{10}, because that would double the forward probability. The inner probability ν_7 of state q_7 however is common to both paths, so it can be used in the calculation.

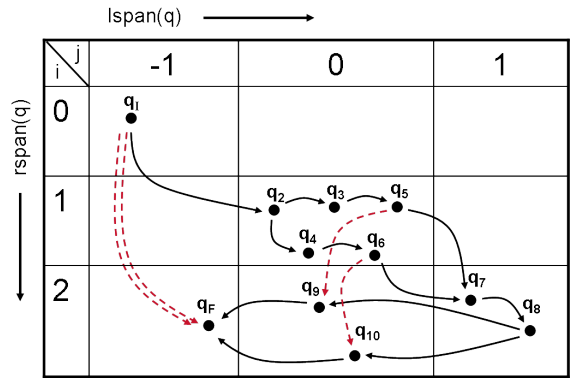


Figure 7.6: States and their transitions in the LCG chart (the q's refer to Table 7.1).

7.1.6 Implementation issues

Retrieving the Viterbi parse

After constructing the chart, the Viterbi parse can be retrieved by following back the Viterbi pointers stored in the chart cells, starting from the final state. Stolcke [1995, p.22] describes a recursive procedure to recover the Viterbi parse for the Earley-style PCFG chart parser (section). The left corner version of the chart parser differs from the PCFG version in that the chart also stores states corresponding to shift transitions. These should however be ignored when retrieving the Viterbi parse, and treated as leaves of the tree. In their place the shifted word is inserted in the parse tree, but no recursive call is made.

Left recursion

In a top-down chart parser left-recursive *predictions* pose a problem when updating string probabilities, because the same state may appear several times within a given derivation, giving rise to so-called prediction loops. In left corner chart parsing the problem is less urgent, because in general left-recursive *projections* do not stay within the same state set but are followed by a *shift* to the next state set. Only *unary* projections could still directly and indirectly result in left recursion. Indeed, upon inspection of the Wall Street Journal one finds several such unary rules, for example $NP \leftarrow NP$ and $S \leftarrow VP$ and $VP \leftarrow S$. To deal with this problem we first remove all reflexive unary rules, such as $NP \leftarrow NP$ from the grammar. The remaining indirect left recursion can be dealt with by precomputing a matrix of probability sums for the reflexive, transitive unit-production relation [Stolcke, 1995, p.16]. An alternative, but only approximate solution is to restrict the number of successive unary projections. In this work the latter option was chosen, with a maximum of 2 successive unit projections.

Prioritized queues

When complete states from a certain state set are attached to a goal state, it is possible that the resulting state ends up in the same state set as the complete state. In this case one must take care that new states are added in a specific order to ensure that all contributions to a state's forward and inner probabilities are summed before that state is used as input to further projections or attachments within the same iteration. To achieve this, the newly derived state is inserted into a so-called prioritized queue [Stolcke, 1995, p.32]. States are ordered in this queue according to their left span index, from high to low, such that states that span a smaller part of the input sentence are completed before they are attached to states that contain them.

Unless the transitive unit projection matrix option is adopted, a similar problem may occur with unit projections (that occur within the same iteration as attachments), because unit projections always result in complete states in the same state set. To ensure that unit projections are processed in the correct order a second prioritized queue is created within the first one. In the latter queue unit projections are ordered according to their 'depth of projection', with a maximum of 2.

Beam search

To prevent the chart from growing too large states can be pruned from the chart. This restricts the search space, but may lead to a sub-optimal Viterbi parse, and under estimation of the inner and forward probabilities. One approach to pruning is to maintain a beam of states of a fixed size throughout the chart. States of a certain state set are ranked according to their forward probabilities, and if the maximum beam size is exceeded the states with the lowest forward probabilities are pruned from the state set. The optimal beam size should be assessed empirically. Again, one should take care that all contributions to a state's forward probability are accumulated before one decides whether to prune the state or not.

7.2 Evaluation of the basic probabilistic LCSG chart parser

The development of the PLCSG chart parser in the previous sections was not a goal in itself, but an intermediate step and base model on top of which an episodic left corner chart parser will be built in the next section. Still it is useful to have an idea of its performance before continuing further, because it provides a baseline for comparison to later versions. The PLCSG parser was evaluated on the standard task of parsing the Wall Street Journal, while measuring labeled precision and

recall. As usual, sections 2-21 were used for training the parser, and section 22 for development. Table 7.2 compares the performance of the PLCSG chart parser implemented in this thesis to that implemented by van Uytsel et al. [2001], and to a standard PCFG implementation with lexical and syntactic smoothing[4] (but note that the other results are on section 23 of WSJ, while my results are on section 22). All reported results are for sentences up to length 40 (ca. 1700 in section 22), which took approximately 1.5 hours to parse.

Parsing model	LR	LP	F
This work	70.4	76.7	73.4
van Uytsel (2001)	79	79	79
PCFG	–	–	78.5

Table 7.2: Comparison of different implementations of the left corner parser with the PCFG parser.

Table 7.2 shows that the van Uytsel parser performs more than 5 percentage points better than my implementation of the PLCSG parser, and in the same range as the PCFG parser. Note however, that van Uytsel et al. [2001] did not implement smoothing (as a consequence his parser could not parse 4% of the sentences), while in this work (as well as in the PCFG) the train sentences were horizontally Markovized, and 2 levels of back-off smoothing were performed ($\lambda = 0.2$ for each level). Thus, the current implementation can parse all sentences, but the smoothing may have an overall negative impact on performance. Further, it should be noted that the scores reported by van Uytsel et al. [2001] were obtained with an advanced submodel of the parser, which included additional conditioning history to calculate the probabilities, whereas my implementation is based on the basic PLCSG probability model.

7.3 The episodic left corner chart parser

This section introduces the *episodic* left corner chart parser, an episodic extension of the probabilistic LCSG chart parser of the previous sections. I will discuss two versions of the episodic left corner chart parser, but I have implemented only the second version: the first version computes the most probable parse from the episodic traces, and the second version computes the shortest derivation, that is the derivation containing the least number of episodic fragments. In both cases training proceeds in the same manner as with the episodic reranker (see section 6.2.3): first, the treebank parses are converted to left corner derivations, and treelets are created for unique left corner productions (with specific register positions) in the treebank derivations (including treelets for shift productions).

[4] I am grateful to Federico Sangati for providing these results.

Then, for every left corner derivation in the treebank, and for every step in the derivation the treelet associated with the rewrite rule is filled with traces that encode the sentence number and position in the derivation (see the detailed algorithm in section 6.2.2).

The treelets filled with traces constitute an episodic grammar; the 'rules' of this grammar are thus no longer symbolic rewrite rules, but objects with a local memory where traces are contained. The treelet keeps track of the next required operation in the queue by means of a local register. The states of the LCSG parser are replaced by *treelet states*, which are of the form

$$q = \{G; X \leftarrow {}_j\lambda \ \bullet_i \ \alpha \ , E_q\} \tag{7.15}$$

Here the register position of the treelet is indicated by the dot, and G is again a goal category included in the state to enforce valid parses. The main difference with the LCSG states is that treelet states are enriched with a set E_q of (activated) traces. When a new state is added for the first time to the chart, all the traces are copied from the 'treelet type' to the treelet state, and receive a certain activation.

7.3.1 Most probable episodic parse

The most probable episodic parse is computed in much the same manner as the Viterbi parse in the LSCG chart parser, except that the base probabilities (P_{shift}, $P_{project}$ and P_{attach}) are not estimated beforehand from the treebank, but computed on the fly, as a function of spreading activation of traces in treelet states. This works as follows (but note that it has not been implemented):

When a new treelet state q is first added to the chart (as a result of a shift, project or attach operation) all traces from the 'treelet type' are copied to the treelet state. Then, for every trace separately, its common history (CH) is updated from a predecessor trace according to the update rule 6.2.3 (section 6.4), which is repeated here for convenience: Let q' be the predecessor state (either a complete state in case of project or attach, or a goal state in case of shift), and let $e_{q',i} = (s_{q',i}, n_{q',i})$ be the i^{th} trace in the predecessor state (associated with treelet $t_{q'}$), and $e_{q,j} = (s_{q,j}, n_{q,j})$ a trace in the current state q (associated with treelet t_q). (As before, s denotes the sentence number in the treebank, and n the position in the derivation.) Then

$$CH(e_{q,j}) = \begin{cases} CH(e_{q',i}) + 1 & \text{if } \exists\ (s_{q',i}, n_{q',i}) \text{ such that} \\ & (s_{q,j} = s_{q',i} \ \wedge \ n_{q,j} = n_{q',i} + 1) \\ 0 & \text{otherwise} \end{cases} \tag{7.16}$$

(i.e., the CH of the trace is incremented by 1 if there is a direct predecessor of the trace in the predecessor state.) When the same state is reached multiple times a weighted average is computed for the *CH* of its traces: Let $CH(e_{q \leftarrow q',j})$ denote

the common history of trace $e_{q,j}$ originating from the path through predecessor state q'. Then

$$CH_{average}(e_{q,j}) = \frac{\sum_{q'} CH(e_{q \leftarrow q',j}) P_{fw}(q')}{\sum_{q'} P_{fw}(q')} \tag{7.17}$$

(The rationale for weighing with the (non-episodic) forward probability is that it is a measure of the number of paths that have converged at q'.) Once the average CH's of the traces in a state are known, one can compute their activation $A(e_{q,j})$ according to Equation 6.1. Now it is straightforward to dynamically update the forward, inner and Viterbi probabilities in the chart, using Equations 7.5 to 7.13. In each of the latter equations the base probability (P_{shift}, $P_{project}$ or P_{attach}) is a function of the current state q, which is given by the relative fraction of traces in q that prefer moving to another treelet t_r (corresponding to a shift, project or attach operation), weighted by their activations. This probability was given in Equation 6.2, which is repeated here:

$$P_{episodic}(r|q) = \frac{\sum_{e_i \in E_q^r} A(e_i)}{\sum_{e_j \in E_q} A(e_j)} \tag{7.18}$$

As before, E_q^r denotes the set of traces in state t_q associated with the current state q that points to treelet t_r, and E_q is the full set of traces in state q.

One should take care that all incoming contributions to the CH's of traces in a state are considered for the averaged before continuing to update the CH's of traces in states further down in the derivation. This can be dealt with by a priority queue, as was explained in section 7.1.6.

There is a complicated issue concerning updating the common histories across a shift operation. This again has to do with non-local dependencies in left corner parsing: when you start a shift a new substructure is created for which it is not known yet which state it will attach to. A possible way around this problem is to keep track of an 'inner' CH, like the inner probability in the non-episodic LCSG parser, that starts to count from CH=0 (for every trace) at the shift operation. Upon an attach the 'inner' CH of traces in the complete state can be added to the total CH of traces in the goal state, as is done with forward probabilities.

The difficulty here is that, unlike the inner and forward probability, the 'inner' CH is not independent of the CH of the goal state. One cannot simply add the inner CH's of traces in the complete state to the CH's of traces in the goal state, because whether they should be added or not depends on whether the traces of the goal state have a successor trace after the shift operation. Can one not make the decision to add or not the 'inner' CH's at attach time, once the goal state is known? Unfortunately, it is not as simple as that. One of the complications is that the CH's are not additive: they can be set to 0 at any time during the shift loop. Things get more complicated if one also takes the average CH of contributing paths. As of writing this section the problem remains unsolved.

Remark about episodic parsing

Given that in the episodic grammar one may reconstruct a parse from sequences of successive traces pointing to treelets, one might wonder why it is actually still necessary to use a parser, rather than simply let the parse of the test sentence be decided by a Markov process between treelets? The reason is that there exist non-local syntactic (tree-)constraints that can not be captured by the Markov process. Without such constraints it could happen that, as soon as one combines treelets from different exemplars, in the resulting derivation a treelet state attaches to a state that has not occurred before in the derivation. The Earley chart structure guarantees that only those successions of treelets are allowed that constitute valid parses.

7.3.2 Shortest derivation parse

While in the *most probable* episodic parse traces receive activation values based on their common histories, in the shortest derivation parse the activation of a trace corresponds to the length of the shortest derivation up to the current state and trace. The Shortest Derivation Length of trace j in state q, abbreviated as $SDL(e_{q,j})$, is measured as the number of distinct episodic fragments that are used in the derivation path up to trace $e_{q,j}$ in state q.

Whereas in the previous chapter (section 6.3.2) I described a straight forward way to compute the shortest derivation of a parse tree in a greedy fashion, here I will develop a non-greedy dynamic programming approach that efficiently computes the shortest derivation in the left corner episodic chart parser. It uses a Viterbi-style algorithm that is very similar to the well-known algorithm for finding the most likely sequence of states in a Markov Model (MM) for language.[5]

Whereas in a MM successive time steps on the trellis correspond to successive words in a sentence, in the episodic grammar successive time steps correspond to successive treelets $< t_0, \ldots, t_n >$ in the derivation (see Figure 7.7). In the episodic grammar there is a distinct 'state' associated with every trace in a treelet, and one can define transition probabilities (or rather, transition costs) between all pairs of traces in treelet t_k and treelet t_{k+1}.

Let $e_{k,i} \equiv \langle s_{k,i}, n_{k,i} \rangle$ be a trace of a stored exemplar derivation, where s_i is the sentence number of the exemplar in the training corpus, and n_i is the position in the exemplar derivation. Define a transition cost C between two traces $\langle s_{k,i}, n_{k,i} \rangle$ and $\langle s_{k+1,j}, n_{k+1,j} \rangle$ of successive treelets t_k and t_{k+1} in a derivation

[5]The shortest derivation chart parser was developed independently to the work of Bansal and Klein [2011] to which it is very much related, although their work is formulated in a very different framework.

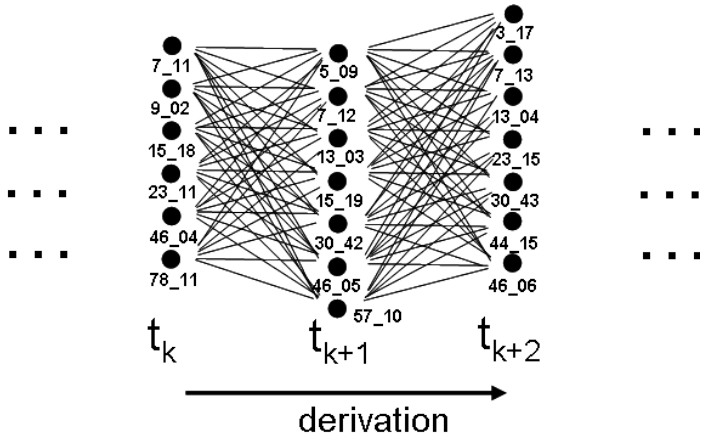


Figure 7.7: Trellis through the traces of the treelets in a derivation

$$C(< s_{k+1,j}, n_{k+1,j} > | < s_{k,i}, n_{k,i} >) = \begin{cases} 0 & s_{k+1,j} = s_{k,i} \ \wedge \ n_{k+1,j} = n_{k,i} + 1 \\ & (\textit{direct successor}) \\ 1 & \textit{otherwise} \end{cases} \quad (7.19)$$

This defines the aggregate 'cost' of any possible path through the trellis (i.e., through a sequence of treelets in a given derivation) in terms of the number of different episodic fragments used. In order to dynamically compute the shortest derivation (the one with the lowest cost), in every trace of treelet t_{k+1} one keeps a pointer to a single trace in the predecessor treelet t_k which, including the current transition cost, follows the path through the trellis with the lowest overall cost. Further, one stores the lowest overall cost (that is, the SDL) thus computed in the current trace. In the final treelet of the derivation one must then find the trace among all traces which has the lowest SDL, and follow the pointers back to the first treelet in the derivation to reconstruct the episodes used in the shortest derivation.

Computing the shortest derivation with discontiguous episodes

The Viterbi-algorithm for finding the shortest derivation can be generalized to the case that discontiguous episodes are allowed. I will briefly describe a possible implementation, but note that it has not been implemented or evaluated thus far, because it is computationally very expensive; the discontiguities give rise to a multiplication of the number of 'trace states' in the treelets for every 'gap' in an episode/fragment that one wants to account for. In the discontiguous shortest derivation case, one may reuse an exemplar that has been used earlier in the derivation, but that has been interrupted by fragment(s) from (an)other exemplar(s), at lower cost than using an entirely new exemplar (provided one

continues the earlier exemplar at a later position n_{k+1} in its derivation than where it was interrupted).

As an example I will describe a 'trigram discontiguity model', which can account for discontiguities that are a result of interrupting an exemplar by fragments from 2 other exemplars at most. In this model a 'trace state' corresponds to the final traces of three distinct exemplars that have been used in the final part of the shortest derivation. (To avoid clutter I left out the trace indices i and j.)

Define $(\mathbf{s}_k, \mathbf{n}_k) = \left\{\left\langle s_k^{-2}, n_k^{-2}\right\rangle, \left\langle s_k^{-1}, n_k^{-1}\right\rangle, \langle s_k^0, n_k^0\rangle\right\}$. Here $\langle s_k^0, n_k^0\rangle$ indicates the final trace of the last exemplar used in the shortest derivation, $\left\langle s_k^{-1}, n_k^{-1}\right\rangle$ indicates the final trace of the second most recently used exemplar, etc. Then one can define a transition cost function which charges a (small) prize for reusing one of the 3 most recently used exemplars in the shortest derivation, and which takes their relative order into account

$$C(\langle s_{k+1}, n_{k+1}\rangle | \langle \mathbf{s}_k, \mathbf{n}_k\rangle) = \\ = \begin{cases} 0 & s_{k+1} = s_k^0 \ \wedge \ n_{k+1} = n_k^0 + 1 & (\textit{direct successor}) \\ 0.1 & s_{k+1} = s_k^0 \ \wedge \ n_{k+1} > n_k^0 + 1 & (\textit{same exemplar w. gap}) \\ 0.2 & s_{k+1} = s_k^{-1} \ \wedge \ n_{k+1} > n_k^{-1} & (\textit{second most recent}) \\ 0.3 & s_{k+1} = s_k^{-2} \ \wedge \ n_{k+1} > n_k^{-2} & (\textit{third most recent}) \\ 1 & \textit{otherwise} & \end{cases} \tag{7.20}$$

As before, for every trace in treelet t_k one selects a single trace state in the previous treelet, which minimizes the overall derivation length, and stores a pointer to it. Subsequently, one updates the states of all traces in treelet t_{k+1}, such that they include the final traces of the 3 most recently used exemplars, and one stores the aggregate cost in the updated states.

7.3.3 Shortest derivation left corner chart parser

Having explained how the Shortest Derivation Length (SDL) is computed *given* a derivation, it is only a small step to implement it in a chart parser. The basic control structure is again the non-episodic (probabilistic) left corner Earley parser, that uses shift, project and attach operation to fill a chart. When moving from trace $e_{q',i} = \langle s_{q',i}, n_{q',i}\rangle$ in predecessor state q' to trace $e_{q,j} = \langle s_{q,j}, n_{q,j}\rangle$ in the new state q, the transition cost $C(e_{q,j}|e_{q',i})$ is computed as in Equation 7.19, replacing the treelets t_{k+1} and t_k by t_q and $t_{q'}$ respectively.

To compute the SDL of trace $e_{q,j}$ one must find the trace $\widehat{e}_{q',i}$ in the predecessor state q' that, including the transition cost to the current trace, yields the shortest derivation length

$$SDL(e_{q,j}) = min_{e_{q',i}}(SDL(e_{q',i}) + C(e_{q,j}|e_{q',i})) \tag{7.21}$$

After this is found, the SDL is stored in trace $e_{q,j}$, together with a pointer to trace $\widehat{e}_{q',i}$ in state q', that will be used to reconstruct the Viterbi path going

through trace $e_{q,j}$. Once the chart is constructed one can reconstruct the shortest derivation parse as usual: in the final state one selects the single trace with the shortest derivation length, and starting from this trace one follows the (trace-level) Viterbi pointers back through the predecessor states and traces until the start state.

Updating the shortest derivation in the chart

In case more than one derivation passes through the same state q (i.e., q has multiple predecessor states q' in the chart), one compares for every trace $e_{q,j}$ in q the SDL of path through the most recent predecessor state q' and trace $e_{q',i}$ with the best *SDL* thus far, as one would do with the Viterbi probability. If the current *SDL* is lower one replaces the *SDL* of trace $e_{q,j}$ as well as its pointer with a pointer to $e_{q',i}$ and state q'. Note that unlike the episodic Viterbi probability, which is associated with the state as a whole, the *SDL* is associated with a trace, hence within a single state distinct traces may point to different predecessor states.

The updates of the *SDL*'s of the traces follow the control structure of the LCSG chart parser described in the previous section; Below I give the update equations for the shift, project and attach operators, where the following notation is used (referring to Equations 7.5 to 7.13): $SDL(e_{\mathcal{N},j})$ is the SDL of the j^{th} trace $e_{\mathcal{N},j}$ in the new state $\mathcal{N}$; $SDL(e_{\mathcal{C},i})$ and $SDL(e_{\mathcal{I},k})$ are similar definitions for traces in the complete state $\mathcal{C}$ and the goal state $\mathcal{I}$; $\texttt{update}(SDL(e_{\mathcal{C},i}), \mathcal{N})$ is an abbreviation of the update Equation 7.21 (involving a transition cost function).

- The *shift* operation (Figure 7.4):

$$SDL_{inn}(e_{\mathcal{N},j}) = 0 \tag{7.22}$$

- The *project* operation (Figure 7.2)

$$SDL_{inn}(e_{\mathcal{N},j}) = \texttt{update}(SDL_{inn}(e_{\mathcal{C},i}), \mathcal{N}) \tag{7.23}$$

- The *attach* operation[6] (Figure 7.3):
  Let $SDL_{min}(\mathcal{I})$ be the length of the shortest derivation to reach any of the traces in the goal state $\mathcal{I}$; Upon attach, the SDL of a trace $e_{\mathcal{N},j}$ in the new state $\mathcal{N}$ is updated according to the equations below

$$SDL_{inn}(e_{\mathcal{N},j}) = SDL_{min}(\mathcal{I}) + \texttt{update}(SDL_{inn}(e_{\mathcal{C},i}), \mathcal{N}) \tag{7.24}$$

[6]Note that this is only an approximation of the actual shortest derivation, because the cost of the shift transition is ignored. See the end of this section for a discussion of the shift problem.

Care should be taken that all incoming paths to a state are considered as candidates for the *SDL*'s of the state's traces before continuing to update *SDL*'s in states further down in the derivation. To this end a priority queue is used, as was explained in section 7.1.6.

Tie breaking

In case of ties with respect to the *SDL* of trace $e_{q,j}$ in state q (i.e., there is more than one predecessor trace that gives rise to the same *SDL*) one can resort to a tie-breaking heuristic, that selects one predecessor trace $e_{q',i}$ in state q' from among all predecessor traces that result in the same SDL. The tie is broken in favor of the trace with the highest Viterbi probability of the path through $e_{q',i}$ and leading up to $e_{q,j}$. Note that within a certain state of the chart every trace has its *own* Viterbi path and Viterbi probability. (Recall that traces from one state may store pointers to different predecessor states, depending on their shortest derivation.) Thus, the Viterbi probability is computed for every trace individually according to the non-episodic LCSG probability model, irrespective of the overall Viterbi path and probability associated with the state.

The Shift problem

As was discussed in section 7.3.1, for the shortest derivation parse, too, there is an issue that the updates of the SDL are complicated by non-local dependencies: the attach operation must operate on a goal state that has been derived earlier in the path of the same derivation. If one would naively update the SDL after every shift, project and attach operation one might end up with a shortest derivation that is not a possible path (namely, by attaching an episodic fragments to a goal state that is not in the path). To deal with this problem one must keep track of an 'inner' SDL, which is set to 0 for every trace upon a shift operation. Upon attach the 'inner' SDL is added to the total SDL of the goal state.

However, still the solution is not complete: if one resets the 'inner' SDL of every trace to 0 upon a shift operation, one ignores the fact that a shift sometimes continues an existing episode, and sometimes starts a new episode. Yet, which case applies is only known at the time of attach, because of the non-local dependencies. Unfortunately, I have thus far not been able to address this problem in an efficient manner. Therefore, the current implementation, pertaining to the results reported in the experimental section, ignores switch costs at the shift transition, hence it cannot be guaranteed to always find the shortest derivation.

7.3.4 Implementation issues of the shortest derivation parser

- **Duplicate sentences are not allowed**. When parsing a new sentence with the shortest derivation, the trivial solution, which uses a parse of

the identical sentence as the sole fragment in the derivation, should be avoided. To this end, before parsing duplicate sentences are removed from the exemplar set.

- **Time complexity**. The time complexity of parsing a sentence with the shortest derivation LCSG parser is $O(n^3XN)$, where n is the sentence length, and N is the number of sentences in the train corpus. The standard left corner chart parser has time complexity $O(n^3)$, because there are in the order of n^2 cells in the chart, and every chart cell can be derived in the order of n ways. The episodic left corner chart parser further iterates over the traces in a state, whose number is in the order of N (if trained on the full WSJ, certain states may contain up to 100,000 traces).

- **Improving the efficiency**. To make the updates of the transition costs in Equation 7.19 more efficient, one may choose to consider as candidate predecessor traces only those traces in a predecessor state that have minimal SDL, since other traces can never give rise to shorter derivations. The 'minimal SDL traces set' (in short *minSDLset*) of a state can be computed beforehand, and this needs to be done only once for every state. Although this substantially improves efficiency, the downside of this option is that it introduces a certain bias because it may miss a candidate trace outside the minSDLset if it happens to be the direct predecessor trace of the current trace (hence has zero transition costs, whereas the traces in the minSDLset have transition cost 1). The latter trace is excluded from tie-breaking even though it may have across the board the same SDL as traces in the minSDLset of the predecessor state. Thus, the heuristic prefers traces in the minSDLset that do a 'late switch' (i.e., to a different fragment) over a trace that has switched earlier.

- **Goal categories in treelets**. Although the treelet *states* are distinguished by a goal category, in theory a goal category should not be included with a treelet *type*. However, empirical tests indicate that adding a goal category to treelets hardly impacts performance, while on the other hand it significantly speeds up the parser (because the same number of traces is distributed over many more treelets).

- **Sampling at the trace level**. If one is interested to introduce some randomness into the system but still preserve the shortest derivation, on may choose to sample among predecessor traces that participate in a tie-break. Sampling in the episodic parser is different than standard sampling because it is done at the trace level, rather than at the state level (recall that every trace defines its own Viterbi probability and path). Sampling will become important in the next chapter when the episodic parser is invoked

in learning, and a certain amount of noise is needed to break the symmetry of the parses.

7.4 Experiments with the shortest derivation left corner parser

The episodic left corner shortest derivation parser (ELCSD) was trained on the Wall Street Journal, sections 2-21, and tested on section 22. Labeled precision and recall was computed for the shortest derivation parses, with 2 levels of probabilistic tie-breaking, as described above. As the research is still in its developmental phase I present only preliminary results; the parser was trained only on the first 50 % of the WSJ train set, and tested only for sentences up to length 20. This took approximately 60 hours for the 709 sentences of section 22. To have a fair baseline, the standard (non-episodic) probabilistic LCSG chart parser was also trained on 50 % of the WSJ. Table 7.3 summarizes the results. (See Table 7.2 for the results of the non-episodic left corner parser trained and tested on the full WSJ.)

Parsing model	LR	LP	F	EM
ELCSD	82.3	81.1	81.7	26.9
SL-DOP [Bod, 2003]	90.7	90.8	90.7	–
AFG [Bansal and Klein, 2011]	–	–	86.9	31.5
PLCSG (baseline)	76.3	81.2	78.7	16.8
van Uytsel (2001)	79	79	79	–

Table 7.3: Comparison of the episodic left corner shortest derivation parser (ELCSD) with state-of-the-art shortest derivation parsers, and with the baseline PLCSG and the van Uytsel left corner parser. AFG= all-fragments grammar; SL-DOP=Simplicity-Likelihood-DOP. Note that all other parsers were tested on section 23, and on sentences up to length 40, while the ELCSD and the baseline were tested on section 22, and sentence length <= 20.

From Table 7.3 it is clear that the ELCSD parser is not (yet) competitive with the state-of-the-art, but it outperforms the baseline PLCSG, and even the more sophisticated implementation of a PLCSG by van Uytsel et al. [2001] by a significant margin. Further, it should be noted that the results of Bansal and Klein [2011] were obtained after a coarse-to-fine pruning preprocessing step, and that their simple shortest derivation implementation scored badly, with $F = 66.2$. The high variance in the F-scores of our results ($\sigma = 18.7$) gives hope that a similar approach could also work for the ELCSD parser (see section 7.4.1).

7.4.1 Coarse-to-fine parsing and pruning

An example from the WSJ illustrates that too heavy a reliance on the shortest derivation for parsing a novel sentence can sometimes have a detrimental effect on the accuracy of the parse. Figure 7.8 (b) shows that the shortest derivation parse has literally copied a fragment from train sentence 11121 containing an analysis of the words *an average* as separate constituents. However, this analysis of *an average* in the shortest derivation parse is highly unusual. As a result it scores much worse (F=0.43) than the standard PLCSG model parse (b), which scores F=1.0.

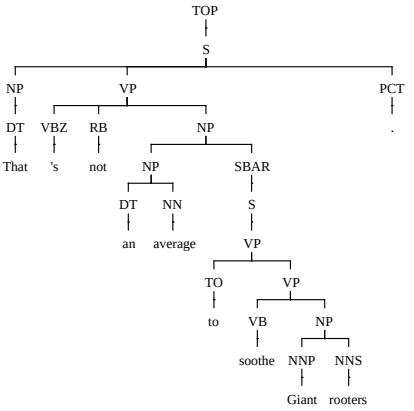
Ideally, one would like to avoid the use of fragments in the shortest derivation that occur only rarely in the train set. From the results of Bansal and Klein [2011] it can be learned that coarse-to-fine parsing has a surprisingly large positive effect on the shortest derivations, because it filters out idiosyncratic exemplars with low frequency. It is expected that pruning the states of the chart by using the forward probability of the LCSG probability model will have a similar effect, and it can be done in a single pass through the data. This is left for future work.

7.4.2 Chapter conclusion and discussion

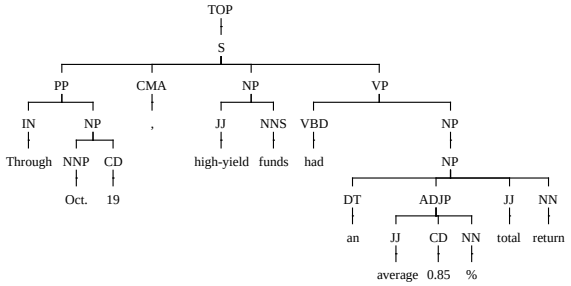
Whereas in the previous chapter the episodic grammar was implemented as a reranking system, meaning that its performance depended on a third party parser, the current chapter demonstrated that episodic parsing is a viable approach in its own right, and computationally tractable. In the preliminary evaluation its performance is lower than state-of-the-art, but there are reasons to be optimistic about the future, in particular if the full power of episodic parsing can be unleashed, once we succeed to solve the problem of the shift transitions. Otherwise, it is currently the best performing left corner parser, and hopefully will restore confidence in left corner parsing as an attractive, cognitively plausible alternative to top-down parsing.

Relation to other work

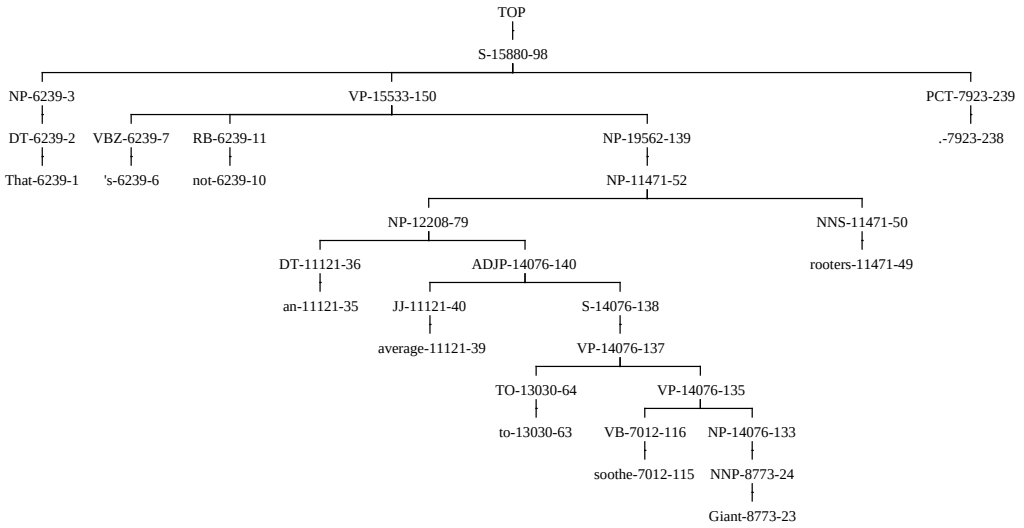
The episodic shortest derivation parser bears a strong resemblance to the work of Bansal and Klein [2011], even though it is formulated in quite a different framework, and it was developed independently from that work. Their All-Fragments Grammar (AFG) uses a technique known as Goodman Reduction [Goodman, 1996], and takes it to its extreme: every local subtree in a parse tree of the train corpus defines a unique rule ($X_p \rightarrow Y_q\ Z_r$), with unique labels. As a result the AFG grammar has very many specific rules, but only 2 general rule schemes: *CONTINUE* ($X_p \rightarrow Y_q\ Z_r$) and *SWITCH* ($X_p \rightarrow X_q$). CONTINUE follows unique rules along training trees, whereas SWITCH changes between trees. In the shortest derivation application, a CONTINUE rule is associated with a transition costs of 0, whereas the SWITCH rule has associated cost 1.



(a) Gold Standard parse



(b) Fragment of train parse tree used in the shortest derivation



(c) Shortest derivation parse

Figure 7.8: Illustration of where the shortest derivation parse goes wrong. (a) Gold Standard parse. (b) Shortest derivation parse. The numbers in the nodes of the tree represent the traces: 11121-39 indicates that this fragment originates from position 39 in the derivation of sentence number 11121 in the train set. (c) Fragment of train parse tree (sentence 11121) used in the shortest derivation

An interesting insight can be obtained if one realizes that the unique local trees of the Goodman reduction in AFG can in fact be interpreted as traces, because they point from one rule in a certain exemplar to a unique successor rule in the same exemplar. For this reason the AFG approach is probably broadly equivalent to the shortest derivation version of the episodic grammar. Yet, the episodic grammar also allows for defining alternative probability models over the traces, such as the probabilistic episodic grammar proposed in section 7.3.1. Further, it is important to note that, by contrast to AFG, episodic-HPN is a processing model: the order of traversal through the path in a derivation is a determining factor in the search for its shortest derivation. Another difference, of course, is that the current work is implemented as a left corner parser, whereas Bansal and Klein [2011] assume a top-down parsing strategy.

Simplicity-DOP, or S-DOP, is the DOP implementation of a shortest derivation parser [e.g., Bod, 2003, 2000]. It works by assigning equal probabilities to all DOP-fragments, large or small, such that the system will exhibit a preference for parsing with a minimal number of fragments. Bod [2003] proposes two variations of the tie-breaking heuristic: Simplicity-Likelihood-DOP, or SL-DOP, selects the shortest derivation tree among the n likeliest trees, while Likelihood-Simplicity-DOP, or LS-DOP, selects the likeliest among the n shortest derivation trees. As Table 7.3 shows, SL-DOP is currently the best performing system among the shortest derivation parsers. For a further comparison between DOP and the episodic grammar framework please refer back to section 6.4.

Interpretation of the episodic grammar as a syntactic network

To conclude this chapter let me bring back in the reader's mind the original motivation behind the episodic grammar agenda, which was to find a solution for syntactic parsing within a connectionist framework. To this end I demonstrated that in the episodic grammar all conditioning events can be accessed locally (i.e., the contextual history is content-addressable through the traces), in line with the constraints of a connectionist design. I mentioned before that the treelets of the episodic grammar fulfill the role of traditional grammar rules. They should be regarded as physical network units, that possess a local memory containing the traces. Further, the treelets possess a local register to keep track of which of their children has last been processed. The entire set-up is compatible with a view of episodic grammar as a physical network, consisting of multiple autonomous treelets that work together to produce the macro-behavior of a syntax, without central control. In this view a treelet is a kind of micro-processor, that locally enforces the correct order of execution of a sequence of operations through the register (i.e., starting with a projection, and followed by zero or more shifts and attachments).

While in the current chapter the treelets were created by copying rules from an annotated treebank in a supervised manner, and identified by symbolic labels,

in the next chapter the labels will be removed as well, as the episodic grammar will be integrated with HPN. The treelets will then be replaced by compressor nodes, and the labels by vectors in a high-dimensional substitution space, which learn their position in the space from experience. Hence the episodic framework generalizes to a *connectionist* syntactic network that implements a left corner parser, and learns in a fully unsupervised manner.

Chapter 8

Learning grammar through episodic memory consolidation

This chapter completes the integration of the episodic grammar with the HPN framework in the episodic-HPN *model. The enrichment of HPN with an episodic memory allows conditioning on sentence history, and at the same time learning from analogy among stored exemplars. The approach is motivated from the usage-based language acquisition literature, which shows that a similar process of analogy extraction drives the discovery of general, productive rules from specific utterances.*

The episodic-HPN model assumes a bi-directional interaction between episodic and semantic memory in syntactic bootstrapping: on the one hand the topology (semantic memory) is used to compute distances between treelets where no episodic memory traces are present (hence, it can deal productively with unseen sentences), while on the other hand the episodic memory is used to find a 'shortest derivation' parse, which results in a gradual adjustment of the topology to episodic experiences. As such, learning a grammar in episodic-HPN parallels the process of memory consolidation and de-contextualization in the brain, whereby an abstract semantic memory is gradually extracted from concrete episodic memories. After presenting the formal model, I discuss how the episodic-HPN framework differs from existing connectionist models of memory consolidation, and I point out that it predicts a role for the hippocampus in dynamic binding.

8.1 Introduction

This chapter introduces the episodic-HPN framework, which integrates an episodic memory with the HPN network. It is based on the *episodic grammar* formalism, which was presented in the previous chapters. By contrast to the episodic grammar, which is a supervised and symbolic system, episodic-HPN learns fully unsupervised and is connectionist. But unlike the 'semantic' HPN model of Chapter 5 it addresses the problem of conditioning on sentence context by using episodic memory.

In this chapter the two-way interaction between the episodic and semantic memory systems will be modeled. While the episodic grammar modeled language processing (parsing) as the retrieval from semantic memory of episodic memories (reconstructed from traces), episodic-HPN additionally models grammar acquisition as a process of memory consolidation from episodic to semantic memory. As discussed in Chapter 2 memory consolidation refers to the gradual construction of a relational semantic network out of episodic memories. This involves a search for structural analogies between episodes in order to infer shared semantic features, resulting in de-contextualization of the episodes.

Grammar acquisition, like memory consolidation, concerns the question of how abstract knowledge (e.g., the rules of a grammar) is extracted from experience (e.g., sentences). The central claim of this chapter is that grammar acquisition parallels the process of memory consolidation, i.e., it must be construed as a process of analogical inference from episodic to semantic memory. To motivate this claim I will review two well-known studies from the language acquisition literature, and I will formulate conditions for successful grammar induction based on an analysis of the BMM algorithm (section 3.2.2). Then I will present the formal model of episodic-HPN, which shows in technical detail how the consolidation process can be implemented in a connectionist network. Unfortunately, at this stage I have not yet been able to evaluate it quantitatively. However, I do asses qualitatively some predictions of the model relating it to the neuro-biology of memory, and to other models of memory consolidation.

8.2 The case for treating language acquisition as a memory consolidation problem

In this section I will review two examples from the literature of grammar acquisition that can be seen as instances of the claim that grammar acquisition, like memory consolidation, involves an analogical inference process based on episodic memories.

8.2.1 Example 1: Lieven et al. [2003]

The first example is a dense corpus study in the usage-based tradition by Lieven et al. [2003], aimed at tracing back the sources of creativity of children's speech. In this study Lieven et al. [2003] showed that most utterances produced in one day by a two year old child could be reduced to utterances produced in the previous 6 weeks by using only a single combinatorial operation. For each target utterance they searched the closest matching utterance produced by the child in the preceding weeks, and analyzed the ways in which the novel utterance differed from it. In particular, the number of operations needed to arrive from the closest match to the target utterance was determined. (Operations were 'substitute', 'add', 'drop', 'insert' and 'rearrange'.) It was found that of the target utterances that had not been said before in their entirety (37 % of the total) 74 % could be composed from previous utterances with a single combinatorial operation.

This result suggest that children vary their speech based on analogy with previous utterances. Analogical learning allows bootstrapping general rules, offering children a gradual path to an abstract grammar.

8.2.2 Example 2: Marcus et al. [1999] and Marcus [2001]

As a second case study consider the problem of generalization of sentences of the form

(8.1) *A rose is a rose.*

(8.2) *A lily is a lily.*

(8.3) *A tulip is a tulip.*

These examples were used by Marcus [2001] to train a Simple Recurrent Network (SRN). According to Marcus [2001, p. 50] the SRN could not generalize this to

(8.4) *A blicket is a ...*

Human infants, on the other hand, are able to make generalizations of this kind. In a much cited experiment Marcus et al. [1999] showed that 7-month-old infants can learn an artificial grammar of the form ABB, ABA or AAB, and generalize these simple patterns to patterns consisting of words they had not heard during the training session. According to Marcus, the reason why infants can and the SRN cannot generalize these examples correctly is that infants apparently possess a learning system that allows them to extract algebra-like rules that represent relationships between variables, such as identity, whereas the associative learning mechanism of the SRN is only sensitive to transitional frequencies.

While Marcus is correct that *distributed* networks cannot learn relations over variables, the work on HPN shows that there exist cognitively plausible connectionist learning algorithms that *can* simultaneously exploit rule-based structure and distributional information (see sections 4.7.3 and 4.8.1). The mere ability to represent relationships between variables (or invariants, as in HPN) however still does not explain how a system (or a child) actually discovers the 'correct' variables (as demonstrated in section 5.5.2, this is not a trivial issue for any learning algorithm).

I propose a different interpretation of the fact that, in contrast to the SRN, humans find it easy to generalize the above examples. According to this interpretation the SRN can only generalize based on *similarity* of the examples, whereas humans generalize based on *analogical* reasoning.Thus it seems that syntax learning exploits a basic cognitive capacity for discovering analogies between the internal representations of stored examples. Finding analogy is a higher order process than finding similarity, because analogy concerns similarity of relations: the discovery procedure requires performing pairwise comparisons between stored internal representations. To do so, a system has to keep track of the representations of all previously processed sentences, which in the SRN, or any other connectionist system without an episodic memory, are lost.

8.2.3 The use of analogy in computational models of grammar induction

To clarify the relation between grammar learning and discovery of analogies further, let us closely examine how a typical grammar induction algorithm, such as Bayesian Model Merging (BMM) [Stolcke and Omohundro, 1994, Stolcke, 1994], would succeed at learning the abstract relation underlying the above examples (for a short discussion of BMM see section 3.2.2). If the BMM algorithm is trained on the following sentences (from Stolcke [1994, p. 83])

$$\begin{array}{ll} a\ a & (10\text{X}) \\ a\ a\ b\ b & (5\text{X}) \\ a\ a\ a\ b\ b\ b & (2\text{X}) \\ a\ a\ a\ a\ b\ b\ b\ b & (1\text{X}) \end{array}$$

then it will initially create a unique rule for every sentence, and unique nonterminal symbols for every occurrence of a and b, as in

$$\begin{array}{rl} S & \rightarrow A1\ B1 \\ & \rightarrow A2\ A3\ B2\ B3 \\ & \rightarrow A4\ A5\ A6\ B4\ B5\ B6 \\ & \rightarrow A7\ A8\ A9\ A10\ B7\ B8\ B9\ B10 \end{array}$$

Subsequently, it will perform a hill-climbing search for an optimal grammar, using *merge* and *chunk* operations to move in the space of possible grammars, while

after every operation it checks if it improves a certain objective function. If this objective function incorporates a preference for smaller grammars, as is the case for the Minimum Description Length (MDL), then it will reward a combination of merges and chunks if that uncovers an *analogy* that is hidden in the data. For instance, after merging all preterminals that rewrite to the same terminal, a subsequent chunk of $(A\ A\ B\ B)$ into the single nonterminal X results in a reduction of the size of the grammar

$$\begin{array}{ll} S & \rightarrow A\ B \\ & \rightarrow X \\ & \rightarrow A\ X\ B \\ & \rightarrow A\ A\ X\ B\ B \\ A & \rightarrow a \\ B & \rightarrow b \end{array}$$

The fact that MDL is able to discover recurring rewrite rules is thus due to the existence of analogical sentence pairs in the data. Finally, X can be merged with S, such that eventually one obtains the recursive grammar

$$\begin{array}{lll} S & \rightarrow A & B \\ S & \rightarrow A & S\ \ B \\ A & \rightarrow a & \\ B & \rightarrow b & \end{array} \tag{8.5}$$

It is important to note that thereby at every step BMM evaluates the objective function *globally*, on the entire corpus. From this analysis one may conclude that two things are necessary for a learning algorithm to find rules by analogy

1. *Simplicity bias.* A learning algorithm will not find the rules of the grammar if it is not somehow forced to find a more compact description of the data. This is possible if there are analogies hidden in the structure of the data; making these explicit as grammar rules reduces the number of rules needed to describe the data, hence the description length.

2. *Episodic memory.* A learning algorithm can only find analogies, or merges, if it tries out comparisons between all pairs of sentences. Therefore, for learning all previously processed data has to be available to the system.

I propose that both conditions must be fullfilled if one wants to build a successful connectionist model of grammar acquisition.

8.2.4 Towards a connectionist model of memory consolidation in language

In essence, Marcus' results, showing that infants can generalize patterns to unseen words, and his subsequent demonstration that the SRN fails at the same task

can be reanalyzed and summarized as follows: language (or rather, grammar) acquisition should best be construed as a process of memory consolidation from an episodic to a semantic memory (and not as mere statistical learning of transition frequencies). The possession of an episodic memory is a necessary condition for a learning strategy that is based on the discovery of analogies, which seems to be the underlying strategy in language learning. The study of Lieven et al. [2003], as well as an analysis of the inner workings of the BMM algorithm point to the same insight.

From this it can be concluded that *connectionist networks that do not have a built-in episodic memory cannot learn a (phrase structure) grammar*, because they lack an ability for analogical inference. This applies specifically to recurrent networks such as the SRN, but also to 'semantic' HPN. Since these networks do not keep analyses of processed sentences, the induction of recursive, context free grammars from examples is theoretically impossible.

8.2.5 Discovering analogies via the principle of the shortest derivation

The analysis of the BMM algorithm in section 8.2.3 suggested that, given an episodic memory, a simplicity bias is still needed to drive learning toward an 'optimal' grammar. How can a preference for a smaller grammar be implemented in the brain (or in a connectionist network)?

A cognitively plausible solution, according to many, is to assume that the brain implements a principle of least cognitive effort, by using the *shortest possible derivation* of a sentence. The *principle of the shortest derivation* has been introduced in Data Oriented Parsing as a way to parse novel sentences in terms of fragments of earlier processed sentences, and as an alternative to probabilistic parsing [e.g., Bod, 2000]. It has also been used for unsupervised grammar induction with U-DOP [Smets, 2010].

By reusing existing fragments as much as possible the grammar is kept at minimal size; hence, this indirectly implements a simplicity bias. A learner that uses the shortest derivation will try to discover and reuse shared structure from examples. For instance, given the sentences from section 8.2.2 (*A rose is a rose*, etc.), it will prefer to reuse a rule such as '*X is X*' rather than idiosyncratic rules in the derivation of new sentences. The study of Lieven et al. [2003] seems to indicate that children use a similar strategy (i.e., a minimal number of edit operations) to produce new sentences.

Parsing with the shortest derivation provides the brain with a tool for discovering analogies from a structurally organized episodic memory space. In general terms it involves a cognitive ability to analyze a new experience in terms of a minimal number of previously analyzed experiences. Presumably the ground work of the memory consolidation process, i.e., finding structure in the daily stream of

episodic experiences, can be traced back to a search for the shortest derivation also in non-linguistic domains. As shown in section 7.3.2, such a search procedure can be executed locally, conforming to the connectionist constraint, provided episodes are encoded as distributed traces in the network units (as proposed by the episodic grammar framework).

In the next section a similar local procedure for finding the shortest derivation (and with it, analogies) will be implemented in a connectionist version of the episodic grammar, episodic-HPN. It is expected that by virtue of a parsing strategy that prefers the shortest derivation an optimal (minimal) grammar can be bootstrapped from plain text, where the 'semantic' version of HPN (without episodic memory) failed (e.g., see the experiment in section 5.5.2).

8.3 The episodic-HPN model

The episodic-HPN grammar is based on the episodic left corner shifting grammar (e-LCSG; see section 7.3). The primitive units of the grammar are treelets containing episodic traces. However, instead of reading off treelets from the CFG rules of a treebank, in episodic-HPN the treelets are based on the compressor nodes and input nodes of the HPN grammar; hence they have no labels. The parser is built on top of the episodic shortest derivation left corner chart parser that was developed in Chapter 7. The shortest derivation parse is selected for the reasons discussed in the previous section. Learning is integrated with parsing: fast, instant learning occurs as episodic traces are added to the treelets involved in a derivation after successfully parsing a sentence. In addition, slow, statistical learning follows the algorithm for updating the representations of units across bindings, adopted from HPN (section 5.4). I will first discuss the initialization of the episodic-HPN grammar, then the parser and then the learning algorithm.

Initialization

When episodic-HPN is initialized, episodic treelets are created for every compressor node in every possible register position.

Definition 7 (HPN compressor node treelet)**.** An HPN compressor node treelet is a triple $\mathcal{T} = \{X, n, E\}$, where X denotes a unique compressor node, n denotes the ordinal number of the *active* slot in X (i.e., the register position), and E is a set of traces from sentences that have visited the treelet, initially empty.

A distinct *shift treelet* is created for every combination of an HPN input node (corresponding to a word) and shift slot of a compressor node (see the section 'Language model' below).

Definition 8 (HPN shift treelet)**.** An HPN shift treelet is a 4-tuple $\mathcal{T} = \{X, n, W, E\}$, where X denotes a unique compressor node, n denotes a *shift*

slot of X, W denotes an input node in HPN corresponding to word w, and E is a set of traces from sentences that have visited the treelet, initially empty.

Parsing

Like in the symbolic case, episodic HPN is implemented as a shortest derivation episodic left corner chart parser. The construction of the chart (involving *shift*, *project* and *attach* operators) is described in section 7.1.3; when a new treelet state is added to the chart all the traces are copied from the 'treelet type' to the treelet state, and receive a certain activation (i.e., a value for the Shortest Derivation Length (SDL)). HPN treelet states are defined in analogy to symbolic treelet states (see section 7.3).

Definition 9 (HPN treelet state)**.** An HPN treelet state q, associated with a treelet $\mathcal{T}$, is a 4-tuple $q = \{\mathcal{T}, i, j, E_q\}$, where $\mathcal{T}$ is either a compressor node treelet or a shift treelet; j is the left span index, i is the right span index, and E_q is a set of *activated* traces. If $\mathcal{T}$ is a shift treelet then $i = j + 1$.

The shortest derivation parser uses two levels of tie-breaking in case of equal derivation length. The first (optional) level of tie-breaking is the probabilistic left corner model, as estimated from the relative frequencies of the sentences processed up to the current point. Note that while for the symbolic, supervised parser relative frequencies are estimated only once, from an annotated treebank, in episodic-HPN the frequency counts are updated after every parse and probabilities have to be re-normalized. This level computes project, attach and shift probabilities conditioned on a left corner and goal category, which in case of HPN are identified with a root vector and goal slot. Initially these probabilities will be zero for most events.

The second level of tie-breaking consists of back-off probabilities computed from the HPN metric (i.e., distances between root and slot vectors in substitution space), which are conditioned on the left corner (i.e., the root of a compressor node or a word unit) alone. The back-off probabilities must again be recalibrated (renormalized) after every parse, because the root and slot vectors of nodes that are involved in the parse may have changed as a result of learning.

Although it may seem as though the HPN metric only contributes in a minor way to the parse decision, as it is is only used for tie-breaking (i.e., for computing back-off probabilities), in fact in the early stages of learning the role of the metric will be dominant, because there are still very few traces (exemplars) to derive the shortest derivation with, and most first level probabilities will be zero. The metric, however, yields non-zero probabilities for all events right from the start because node representations are initialized with random values.

Learning

Learning in unsupervised episodic HPN occurs after every parse, and involves the following steps:

1. The shortest derivation of the sentence is found, or in case of ties the most probable shortest derivation.[1]

2. Fast, one-shot learning: Traces for the current sentence are stored in the treelets along the derivational path of the winning parse (episodic memory consolidation).

3. Slow, semantic learning: Vector representations are updated for compressor nodes and word nodes that participated in the winning parse, as described in section 5.4; back-off probabilities are renormalized from the updated metric.

4. (optionally) Frequency counts for project, attach and shift transitions are updated, and project, shift and attach probabilities are renormalized.

Language model

In order to cope with one of the limitations of HPN identified in section 5.6.3, namely its inability to compute string probabilities, HPN was modified in a way that enables it to represent and learn shift transitions to a word. To this end, compressor nodes are equipped with an extra set of *shift slots*, apart from the regular slots, which are invoked at every shift operation. (Thus, there is one shift slot between every two regular slots.) Input nodes in episodic-HPN also have a slot, unlike in the previous version of HPN, where the shift slots of the compressor nodes bind to. Thus, in episodic-HPN the input nodes are not actually terminal nodes anymore. During learning, the vector representations of shift slots and input node slots that were involved in the shift bindings of the most probable parse are updated, just like the regular roots and slots in the earlier version of HPN. This means that a complementary, independent substitution space is needed: the *shift space*. Its dimensionality equals the number of words in the lexicon.

Implementation

I have fully implemented the episodic-HPN model to confirm that the described components together constitute a complete model. *Quantitative* evaluation and optimizing the parameters of the model for its use in typical computational linguistics tasks is left for future work. However, we can already evaluate *qualitatively* the predictions the model makes for cognitive neuroscience.

[1] As before, it should be taken care of that previously processed sentences that are identical to the currently processed sentence are excluded from participating in the shortest derivation.

8.4 Predictions of episodic-HPN for memory consolidation

8.4.1 What episodic-HPN says about the transformation (de-contextualization) from episodic to semantic memory

An important contribution of the episodic-HPN framework is that it focuses attention on the parallels between language acquisition and memory consolidation, as it implements a model of grammar induction that is casted in terms of a transfer of linguistic knowledge from specific episodic representations to abstract semantic representations.

To recapitulate, in episodic-HPN a derivation of a processed sentence is instantly stored in the form of episodic memory traces distributed over semantic network units (one-shot learning). These traces are then recruited to compute the shortest derivation in subsequent processing of novel sentences. Thereby they may force a preference for certain parses that are compatible with previous experiences (exemplar-based processing). As a consequence the bindings of network units that participate in the preferred parse are strengthened, resulting in adjustment of their representations, which in turn affects the topological organization of the network. The topology is important for dealing with unseen events in a productive way.

As this demonstrates, the formation of a topology of syntactic categories (the so-called substitution space) is strongly influenced by the episodic shortest derivations, because the latter are a major factor in determining the selected parse for which the root-slot bindings are updated. In terms of memory consolidation this interaction represents a de-contextualization process: it shows how a (context free) semantic memory (the topology) is gradually shaped from contextually bound episodic experiences, until eventually it comes to reflect an individual's personal (linguistic) experience in the form of an abstract grammar.

The proposed approach to memory consolidation, and its neural interpretation depart drastically from previous computational models of memory consolidation [e.g., McClelland et al., 1995, O'Reilly and Rudy, 2001]. In section 8.5.1 I will discuss these models, and contrast them with the current proposal.

8.4.2 The role of the hippocampus according to episodic-HPN

As discussed in section 2.7.1, in the neuroscience literature a special role is reserved for the hippocampus in memory consolidation. There are two aspects of hippocampal function that are intimately related [Eichenbaum, 2004]:

- First, the hippocampus is involved in episodic memory encoding and consolidation. For encoding it binds sequences of discontiguous semantic elements into episodes, which are structurally organized in so-called 'relational networks'; consolidation involves replay of episodic sequences, thereby strengthening semantic relations.
- Secondly, the hippocampus is involved in processing novel configurations by flexible association of semantic elements, that are shared among episodes through the relational networks.

Dynamic encoding of episodic memories in episodic-HPN

Let us first consider how episodic memory replay is accounted for in episodic-HPN. In the episodic grammar framework it was assumed that episodes can be reconstructed from sequences of traces that are encoded with follow-up numbers. However, the question of how successive traces within a derivation are localized in the cortex was for that moment ignored.

A similar problem, concerning how the brain can efficiently recover the address of a unit where a 'tag' is stored in a pending derivation, was addressed in HPN. There the solution involved a switchboard construction that implements an addressor system, as part of the *dynamic binding* approach (see section 4.8.2). The same solution can be adopted for episodic memory replay from traces.

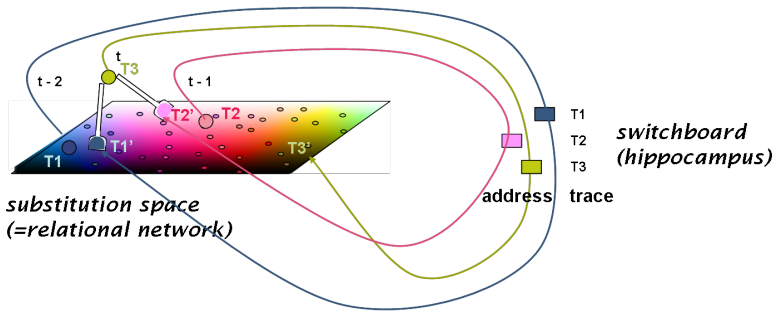


Figure 8.1: Replay of an episodic memory by the hippocampus. When a unit in the substitution space is activated both its address (color coded) and a trace are retrieved. As in dynamic binding, the address is serially transmitted to the switchboard, which tries to match it to a slot. The trace information is used to filter for slots with a matching trace. $\langle T1, T1' \rangle$, $\langle T2, T2' \rangle$ and $\langle T3, T3' \rangle$ are pairs of identical traces stored inside slots and bound semantic units.

Figure 8.1 repeats a simplified version of Figure 5.10 from section 5.7, to illustrate how episodic memories are replayed from their traces. Suppose that trace $T1$ is primed (e.g., by the first word of a sentence) then, as explained in section 5.7, its topological address (encoded in the root of the unit) is serially transmitted to the switchboard, which projects to a neighborhood in the substitution space

where the matching slot (with trace $T1'$) can be found (assuming $T1'$ and $T1$ are near).[2] The same procedure is repeated for the successor trace $T2$, which is linked via the compressor node, etc., until all the original bindings of the episode have been restored.

The hippocampus implements a switchboard function

If, as predicted by episodic-HPN, episodic memories are encoded as distributed traces that are dynamically bound, then that explains that a single system, the hippocampus, is responsible for both replay of stored episodes and flexible association of semantic elements in processing novel events, because both functions involve dynamic binding. This would imply that the hippocampus implements a switchboard function, and that is indeed consistent with the fact that it is situated at the central 'gateway' of the brain: the switchboard must be connected to semantic elements that are distributed throughout the entire cortex in order to be able to dynamically bind them.

From this perspective, *encoding* an episodic memory amounts to making the temporary tags, that are involved in dynamic binding of an event, persistent as episodic memory traces. (Recall from section 5.7 that a critical component of the switchboard solution for dynamic binding is a tagging system, whose function is to attach a unique 'tag' to the units that participate in a binding, such that the bindings are kept 'alive' for some time in working memory.) Specifically, in episodic-HPN the temporary tags that bind the most probable derivation of a sentence are turned into traces (of a stored derivation) by converting them from short term into long term memories.

This idea is consistent with recent neuro-biological findings on memory consolidation, concerning the 'illegibility' of synapses for long-term potentiation [e.g., Izhikevich, 2007]. For instance, according to the 'synaptic tagging and capture hypothesis' [Redondo and Morris, 2010] long-term memory potentiation follows a two-step mechanism, whereby in the first step a so-called 'tagged state' is induced that only creates the *potential* for a lasting change in synaptic efficacy.

The HPN substitution space is an instance of a 'relational network'

According to [Eichenbaum, 2004, e.g.,] the hippocampus structurally organizes episodic memories in 'relational networks', by linking them through semantic elements of episodes that share the same context (see Figure 2.7 in section 2.7.1). Such an organization allows for transitive inference through flexible combination of episodes (which explains why the hippocampus is needed for novel problem

[2] An alternative solution is that within the switchboard an episodic archive is stored, consisting of a list of addresses indexed by episodic trace numbers. In that case the address of a matching trace $T1'$ can be found by querying the list with $T1$, even if the topological locations of the units where $T1$ and $T1'$ are stored are remote.

solving). An example of a relational network is the navigational (cognitive) map, found in the hippocampal place cells of rats, which enables them to navigate their way in a maze (see [e.g., O'Keefe and Nadel, 1979] and Figure 2.8).

The *substitution space* of episodic-HPN, in which the units are enriched with episodic traces, can be regarded as an instance of a relational network, or navigational map, for the language domain. Like relational networks, it organizes episodes, distributed as traces, in a structured memory space, and links them by shared semantic units (containing multiple traces). This allows priming and flexible association with other episodes, allowing for productive use of language (as exemplified in the episodic grammar). Transitive inference (e.g., for finding new routes in a navigational map) is operationalized through the topology of the substitution space. Thus, like rats learning to navigate a maze by structuring spatial episodes (recall Figure 2.8), language learners construct a 'navigational' (topological) map of language by reorganizing episodic linguistic experiences. This process is modeled by episodic-HPN.

Further, episodic-HPN supports at the implementational level the claim of Eichenbaum [2004] that structuring of episodes in relational networks is instrumental in memory consolidation. In section 8.4.1 I explained that the topology of the substitution space (i.e., a semantic memory) is shaped from episodic memory as a result of selecting the shortest derivation parses, and adjusting the topology accordingly.

8.5 Discussion

8.5.1 Relation to other neural network models of memory consolidation

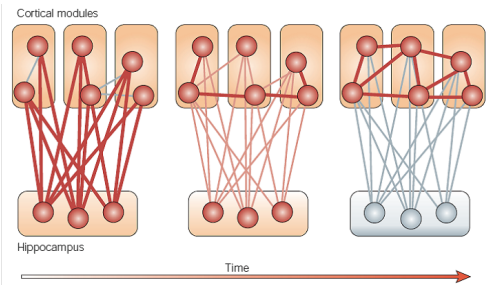


Figure 8.2: Standard consolidation model. The top layer shows the cortical modules containing distributed episodic representations. These are linked to sparse representations in the hippocampus in the bottom layer. (From Frankland and Bontempi [2005].)

Since the current work purports to be a domain general model of memory

consolidation, it is interesting to place it in the context of other modeling work in this field. In the computational neuroscience literature most modeling work on memory consolidation is based on one of two major theories [e.g., see the review by Frankland and Bontempi, 2005]: one is the so-called *standard consolidation model* [e.g., Squire and Alvarez, 1995], and the other is called the *multiple trace model* [e.g., Nadel and Moscovitch, 1997]. According to the standard consolidation model (see Figure 8.2), episodic memories are initially stored in connections between the hippocampus and the cortex. As the hippocampus *replays* the memories (presumably during sleep), the cortico-cortical connections are strengthened, while the dependency on the hippocampus is gradually diminished. This accords with retrograde amnesia studies that show that after a hippocampal lesion, or stroke, episodic memories are lost retro-actively, but memories from a long time before the accident are often preserved, apparently because those have moved out of the hippocampus.

Computational models of memory consolidation in this tradition typically hypothesize that the neocortex and hippocampus act as 'two complementary learning systems' [e.g., McClelland et al., 1995, Battaglia and Pennartz, 2011, O'Reilly and Norman, 2002, Tse et al., 2007]: while the hippocampal system is a fast learner, which stores episodes in one shot as they are processed, the neocortex is a slow learner, which gradually assimilates the episodic experiences within a semantic memory system that represents general, statistical knowledge. This division of tasks is usually motivated by the argument that the different requirements that the human memory system has to cope with, namely learning specifics about the environment (i.e., episodic memory) versus extracting generalities (i.e., semantic memory) are apparently mutually incompatible. According to O'Reilly and Rudy [2001] a single representation cannot simultaneously capture both generalities and specifics, nor can a single learning system combine slow, statistical (integrative) learning with fast automatic recording. In the same vein McClelland et al. [1995] motivate the complementary learning systems approach from the problem of 'catastrophic interference' — this is the phenomenon that, beyond a certain threshold, old memories are overwritten by new ones, which is a known problem for parallel distributed neural networks.

To deal with the trade-off between rapid learning of episodic events and slow learning of statistical structure O'Reilly and Rudy [2001], O'Reilly and Norman [2002] propose a modular network architecture, consisting of several hippocampal and cortical networks (schematically illustrated in Figure 8.2). While the hippocampal networks form sparse representations of episodes (allowing for pattern separation), neurons in the hippocampus are conjunctively bound to distributed representations of the same episodes in the cortical networks (allowing for pattern completion).

In the second tradition, the multiple trace theory (MTT) [Nadel and Moscovitch, 1997] holds, in contrast to standard consolidation theory, that episodic memory traces remain in the hippocampus forever. According to MTT each time an

episodic memory is reactivated this happens in an different context, and consequently a new memory trace is created, with overlapping features in the neocortex, but with a distinctive pattern in the hippocampus, where 'context' is encoded. As a result memories that are often reactivated are associated with a larger number of traces, hence can be retrieved from multiple cues and become more stable. Neural network models in this tradition also assume a modular approach, in which episodic memories are encoded in conjunctive connections between a hippocampal module (with sparse encoding), and a distributed pattern in the neocortical network module [e.g., Nadel et al., 2000]. In this respect also the MTT can be regarded as an instance of the 'complementary learning systems' approach.

Critique of the complementary learning systems approach

A challenge for the complementary learning systems framework is the massive number of connections between the hippocampus and the cortex that result from conjunctive coding of episodes, since every new episode (many thousands a day) recruits at least one dedicated binding neuron in the hippocampus, and must establish connections to the cortex.

Another, more fundamental problem for conjunctive binding of episodes is that it lacks the flexibility to process unseen events by associative expression of stored memories. Yet, as discussed in section 2.7.1, this ability has been suggested by Eichenbaum and Cohen [2001] to be the driving force behind memory consolidation. Hummel et al. [2004] argue that a major limitation of conjunctive coding is that it does not afford to make relational inferences (nor to generalize) beyond specific stored role-filler conjunctions, because conjunctive coding represents all elements of the binding as a single, indivisible entity.

A related problem is that conjunctive bindings provide no temporal or hierarchical structure to episodes. Yet, structure is required for instance to encode causal relations (used for predictions) [e.g., Eichenbaum and Fortin, 2009], syntactic and linear precedence relations in language, or to distinguish the different relations between roles and participants in an event [e.g., Shastri, 2002].

Comparison to episodic-HPN: efficient and flexible encoding of episodes using dynamic binding

The view of memory consolidation that is implied by episodic-HPN differs in important respects from the complementary learning systems approach, because in episodic-HPN episodic and semantic memory are fully integrated within a single system. The learning algorithm of episodic-HPN demonstrates that, contrary the claim of O'Reilly and Rudy [2001], the requirements of fast capture of specific detail versus slow, statistical learning of generalities are *not* incompatible, but can be satisfied simultaneously. Yet, this trade-off was in fact the main motivation for having two complementary learning systems.

The argument of McClelland et al. [1995], that the hippocampus is needed as a buffer in a complementary learning system to prevent catastrophic interference, does not hold either. This argument only makes sense if one assumes that the neocortex is a parallel distributed neural network, yet in this thesis I have defended a localist network view of the neocortex. According to the episodic-HPN model of memory consolidation the reason that the hippocampus is involved as a 'gateway' for episodic memory encoding is not to prevent catastrophic inference, but for its role in dynamic binding, as explained in section 8.4.2.

One of the advantages of dynamically binding episodes is that it saves dedicated binding neurons and connections, hence it does not suffer from the 'massiveness of the binding' problem. Whereas in the conjunctive binding approach every individual episode requires hardwired connections between the hippocampus and the cortex, in the dynamic binding approach of episodic-HPN the same hippocampal element (that is, a single unit in 'substitution space') can be shared by all episodes, and a single projection from this element to the cortex suffices for activating the sensory circuits associated with the element. Internal connections of episodes are handled by the switchboard and the traces.

Further, dynamic binding of episodes in episodic-HPN is responsible for 'flexible expression of memories', which is a condition for the ability to make analogical inference in support of memory consolidation.

Chapter 9

General discussion and conclusions

9.1 Summary

Since this work has covered a lot of information, let me try to briefly reconstruct the structure of the arguments as they were developed in this thesis. The first chapter opened with the presentation of the Memory Prediction Framework as an 'engineering perspective' on hierarchical information processing in the cortex, and the observation that the framework offers very promising prospects for language processing. I pointed out the striking parallel between the solution, proposed by [Hawkins and Blakeslee, 2004], for hierarchical and temporal object recognition in the cortex on the one hand, and syntagmatic and paradigmatic processes in syntax induction, as used in computational linguistics, on the other hand. Yet, I made it clear that the framework does not address several issues that are fundamental in linguistics, such as the productivity of language, an ability for (recursive) substitution of the primitives of a grammar into compound structures or parse trees, and an ability to store and reuse idiomatic expressions or larger fragments/ constructions. To address the issue of productivity I proposed that the cortex adopts its solution for dynamic visual contour binding, attention-mediated and serial spreading of a label, to the language domain in order to dynamically bind syntactic neural assemblies into compound syntactic structures. Further, I discussed the role of topology for encoding graded syntactic category membership in the cortex. Together with the MPF, these ideas formed a neural theory of (the structure of) language that meets Jackendoff's [2002] challenges (see section 9.3).

The MPF suggests a departure from the standard assumptions of (distributed) connectionism, as it proposes that the cortex stores information locally in its columns through hierarchical temporal compression, and that the cortex creates invariance, or abstraction by passing a 'name' between cortical columns (i.e., through *encapsulation* and the use of *pointers*). However, proponents of distributed connectionism maintain that for instance the SRN is capable of learning constituent structure, and systematically generalizes, despite the fact that it does

not explicitly represent hierarchical structure.

To combat these views, in Chapter 4 I tried to sharpen the criteria for systematic generalization that cognitively plausible models of language processing must satisfy. A precise formulation of these criteria made it clear that the systematicity requirement is in fact nothing else than the affirmation of the fact that language has a syntax (that is, a part of linguistic behavior is governed by structure alone), which is (at least) context free: the systematicity criteria can thus be identified as *context invariance* and a *minimum context free inductive bias*. I concluded that the SRN fails both criteria, and that objections (i.e., 'leaky recursion') made against the context free requirement [by e.g., Christiansen and Chater, 1999] do not apply if one approaches the question from the perspective of language learning and generalization.

In Chapter 5 I translated the concepts of the neural theory of syntax proposed earlier into a connectionist model, the 'semantic' HPN model. HPN satisfies the context invariance criterion by virtue of *encapsulation* of its units: input and output of an HPN unit are functionally separated, and units bind using pointers stored in local unit memories. Further, HPN implements a context free inductive bias by virtue of the compressor nodes, which are responsible for hierarchical temporal integration. Although it has many interesting properties, I noted that the expressivity of semantic HPN, based on metric distances between units, is too limited for natural language processing, because it cannot represent contextual information.

I proposed that neural assemblies in the cortex can retrieve contextual information because they have access to an episodic memory. Moreover, episodic sentence memories are content addressable (via semantic memory units), hence locally available, in the form of distributed traces of processed sentences. Thus, it is possible to build a connectionist solution for contextual conditioning by extending the HPN network with an episodic memory.

Chapters 6 and 7 worked out the case of a *symbolic* grammar (trained on sentences annotated with symbolic category labels) with a built-in episodic memory, *episodic grammar*. Exemplar-based parsing with this episodic grammar can be interpreted as episodic memory retrieval, or a priming effect. I showed how the episodic grammar is trained, and how episodic probabilities are computed on-the-fly by reconstructing episodic fragments ('common histories') from the memory traces at parse time. The probabilistic episodic model was evaluated as a reranker of third party syntactic parses on the Wall Street Journal, and its performance is competitive with the state-of-the-art. In Chapter 7 I developed a standalone episodic left corner chart parser, based on an existing left corner chart parser that can compute string (and prefix) probabilities. The shortest derivation version of the episodic left corner parser outperformed the non-episodic left corner parser by a significant margin.

In Chapter 8 the episodic grammar was integrated with HPN, which thus becomes a model of semantic-episodic memory interaction, and the first exemplar-

based connectionist model of language processing and parsing. The content-addressability of episodic traces in the network units allows episodic-HPN to make use of sentence history, while the metric/topology of HPN is responsible for productivity and smoothing, and gradually (under influence of episodic memory-based parses) self-organizes into an abstract, adult grammar. I pointed out an interesting parallel between memory consolidation — the transition and decontextualization from episodic to semantic memory — and language acquisition, in particular from the usage-based or item-based learning perspective. This describes how children's language evolves from concrete utterances and imitation (episodic memory) to progressively more abstract and productive language (semantic memory, i.e., the HPN topology).

Finally, by virtue of an episodic memory episodic-HPN can implement the idea that language is learned by analogy from minimal difference sentence pairs. I argued that the shortest derivation episodic parser does exactly this — learning by analogy — and moreover it serves as a cognitive principle for implementing a simplicity bias: a preference for smaller grammars.

9.2 A possible role for HPN in cognitive linguistic research

9.2.1 Modeling language acquisition with HPN

Episodic-HPN offers a promising framework for modeling language acquisition from a usage-based perspective. The learning process is modeled as the gradual development of a grammatical network (a *constructicon*) consisting of a topology and integrated episodic memory, which evolves from a concrete item-based grammar (episodic memory) to an abstract, adult grammar (semantic memory).

What is important is that, in contrast to the standard distributed connectionist approach, in HPN human language learning (as well as learning in other domains) is seen as a process of categorization, and it is explicitly modeled how during the categorization process syntactic categories are incrementally acquired and become more abstract and adult-like (e.g., see section 5.4.1). Because syntactic categories exist within a continuous topological space, the tricky problem of how to bootstrap them from scratch, a problem encountered by a discrete and set-theoretic notion of categories, is avoided.

Thus, episodic-HPN accords very well with the Usage-Based Grammar (UBG) account of language acquisition [e.g., Tomasello, 2005]. In contrast to the generative all-or-nothing approach to acquisition, UBG can potentially answer the question "how children get from here to there", because it describes the learning process as an incremental route during which children gradually acquire an inventory of 'constructions', from simple and concrete to complex and abstract. In this process children's language use develops from imitation of complete utterances

(the holophrastic phase), through the use of constructions with variable 'slots', to a productive language with abstract, adult-like rules. The same approach is taken by episodic-HPN, where initially the grammar is dominated by concrete, non-productive episodes, which are gradually broken down into fragments, and which guide the formation of an abstract, and productive topology.

The *slots* of compressor nodes in HPN quite literally function as the variable slots in usage-based constructions, like *Where's the X?* and *I wanna X*. Also in HPN a slot becomes progressively more abstract if the input varies much at the slot position, as was illustrated in the example of section 5.4.1: while before learning the slot matches perhaps only a single node, after learning an entire region of nodes (words of phrases) can be substituted at the position of the slot.

9.2.2 HPN and construction grammar

The HPN model shares many of its fundamental assumptions with (Radical) Construction Grammar (RCG) [e.g., Croft, 2001]. As in RCG, in HPN constructions are primary (i.e., a fixed number of (blank) compressor nodes is innately given in the HPN architecture), and syntactic categories (i.e., the substitution space) are derived relative to constructions and have local validity or scope only. HPN shows that the local scope of a syntactic category (i.e., an HPN unit) is acquired through usage, namely when the unit binds to slots in a parse of a sentence.

I believe that the episodic-HPN network is a suitable candidate for a computational model of a *constructicon*, as it encodes paradigmatic and syntagmatic relations between syntactic categories and constructions within a network, as well as idiomatic expressions (episodes). Moreover, HPN shows how the constructions are gradually integrated in the network through experience. The episodic-HPN model further shows that the constructicon can be interpreted in terms of an integrated semantic-episodic memory system.

Obviously, in the HPN framework constructions lack a semantic pole, hence HPN does not model the syntax-semantics interface, which is at the core of the linguistic theory of Construction Grammar [e.g., Goldberg, 2003, 2006, Bergen and Chang, 2005]. However, as was argued in section 2.9, this is not a fundamental design problem. HPN was modeled in part after the Memory Prediction Framework, which is primarily tailored to model visual processing [George and Hawkins, 2005, 2009]. Like other connectionist models, HPN can be adapted to meet the requirements of other domains than language, as it proposes a general, non-domain-specific framework for hierarchical processing (except perhaps for linear word order preservation which is specific for language). Thus, HPN can in theory implement a network that integrates inputs from multiple domains (visual, auditory, sensory, language), offering a more formal definition than usual of constructions as 'associations between a syntactic and a semantic pole' within a grounded, embodied view of semantics.

9.3 A reply to Jackendoff's challenges

In this thesis I have proposed a neural theory of syntax (section 2.8), and a computational model based on this theory. This section aims to show how the neural theory meets the four major challenges posed by Jackendoff [2002, pp.58-67]. (For a summary of the challenges please refer back to section 1.2.)

The problem of variables

The commonly accepted justification of the claim that a neural theory of language must be able to represent variables is that operations over variables are thought indispensable as formal constructs to explain the systematicity of language, or syntax [Fodor and Pylyshyn, 1988] (see section 4.2). Variables with *global* scope cannot be represented in the brain or in connectionist models, because they are symbolic (see the discussion in section 4.2.1). Yet, there is no impediment for connectionist models to represent invariant and abstract, prototypical categories with *local* scope. The brain works with invariant representations for high level object recognition, and many studies show that neurons respond to certain abstract features invariantly across different modes of presentation of the stimulus (section 2.1.1). According to Hawkins and Blakeslee [2004] invariant categories are extracted between columns in successively higher levels of the cortical hierarchy by a mechanism of passing an activation-independent 'name' (representing the identity of the column) (section 2.1.3). This mechanism, which I called *encapsulation*, is responsible for creating abstraction in the brain.

On the other hand in section 4.5.2 I identified *context invariance* as one of two criteria that are responsible for the systematicity of language. While symbolic systems satisfy this criterion because the use of variables allows for making context indepence assumptions, I argued in section 4.8 that context invariance can also be achieved in connectionist systems if one allows encapsulation of the network states. This means that a functional separation must be enforced between the input and output activation of network units. In that case one can define a mode of connectivity between the network units that does not depend on their neuronal activation values, but on their intrinsic representations alone. Such a mode of connectivity is offered by dynamic binding, via the switchboard mechanism described in section 5.7. A connectionist system that combines encapsulation and dynamic binding emulates exactly that property of variables that gives systematicity, without using variables.

In the HPN model the slots of the compressor nodes fulfill the role of 'variables'; they function as placeholders where active units (the tokens) are bound. As explained in section 4.8.1, it is more precise to characterize them as *invariants*, because unlike symbolic variables they have no predetermined meaning and global scope. The slot representations, and hence their extensional meaning and scope are learned incrementally through interaction with the environment.

The massiveness of the binding problem

Jackendoff's claim that the number of bindings required for language processing is too massive for a connectionist network to deal with relies on an implicit assumption that the human language system uses *conjunctive* bindings to represent every potential construction or sentence of a language (see the discussion in [van der Velde and de Kamps, 2006]). However, with *dynamic* binding it is not necessary to have dedicated neurons with prewired connections for all possible linguistic constructions; a connectionist system capable of dynamic binding needs only to represent the primitive elements of a grammar in the network units, which can then combine dynamically to form sentences, like the primitives of a symbolic grammar. As explained in section 5.7, in the switchboard solution of dynamic binding every network unit needs to be physically connected only once to a central hub in the network.

The asymmetry between binding in working memory and binding in long term memory

The duality between 'transient' bindings (as in 'lift the shovel') and the bindings of idioms (as in 'kick the bucket') that seem to be stored in the lexicon can be explained by the interaction between two different memory systems, semantic and episodic memory, in language processing. This was the subject of Chapter 6. In short, 'transient' bindings are dealt with by a semantic memory system using dynamic binding between network units. Idioms or fixed multi-word constructions are stored in the form of an episodic sentence memory, which consists of (numbered) traces distributed inside the semantic network units that participated in the derivation of the idiom. By following the traces the original derivation of the idiom can be reconstructed.

The problem of two

Jackendoff's assumption that whenever a word occurs twice or more within a sentence it requires more than a single representation in the brain is incompatible with the view of hierarchical and serial processing in the cortex advocated throughout this thesis. The 'problem of two' seems to hinge on the presupposition that the brain conjunctively binds a 'word unit' for the entire duration of a parse, or otherwise processes all syntactic elements of a parse in parallel. However it has been argued (in section 2.5) that language processing in the brain makes use of dynamic and *serial* binding, such that 'word units' are freed up after they have been used once in a derivation.

Nevertheless, all bindings of a derivation must be maintained in working memory for as long as the derivation process lasts. I have argued in sections 2.5 and 5.7 that for this purpose the neural assemblies representing syntactic categories must be equipped with local memories, where 'tags' are temporarily stored. The

possession of a distinctive tag induces a unique 'state' in the neural assembly (see section 5.7). That way multiple occurrences of the same word or syntactic category within a sentence can be distinguished.

If, while processing a sentence, it is consciously experienced as a single entity, then this is only an illusion. In Hawkins' words,

> Because of the hierarchy of the cortex you are able to know that you are at home, in your living room, looking at a window, even though at that moment your eyes happen to be fixated on a window latch. Higher regions of cortex are maintaining a representation of your home, while lower regions are representing rooms, and still lower regions are looking at a window.
>
> Similarly, the hierarchy allows you to know you are listening to both a song and an album of music, even though at any point in time you are hearing only one note, which on its own tells you next to nothing. Higher regions of your cortex are keeping track of the big picture, while lower areas are actively dealing with the fast-changing, small details. [Hawkins and Blakeslee, 2004, p.127]

9.4 Relation to other work on unsupervised grammar induction

Before I start discussing the the qualities of episodic-HPN in comparison to other grammar induction systems, I should perhaps note that at present HPN is not (yet) operational as a grammar induction system on a learning task with realistic data. Hence, the following discussion should be taken with the necessary reservations.

On the other hand it should be said that, contrary to the other learning algorithms I will discuss next, episodic-HPN was designed to meet the demands of a cognitive system, and comply with the constraints of connectionism: all interactions and learning procedures in HPN are local, and no innate categories or labels are assumed.

9.4.1 HPN versus Inside-Outside

A well-known and widely used technique for estimating the hidden parameters of a PCFG from text is the Inside-Outside (I-O) algorithm [Lari and Young, 1990], which is an instantiation of the Expectation-Maximization (EM) algorithm (see section 3.2.1). Computational linguists might wonder whether everything that HPN can do, EM, or the Inside-Outside algorithm can do as well.

It can be argued that one may implement HPN in an EM-setting by simply replacing every slot of a HPN compressor node by a unique non-terminal of a

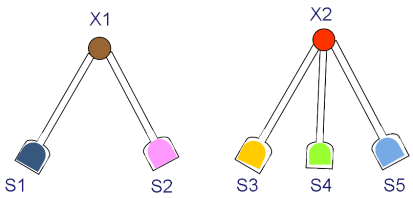


Figure 9.1: HPN network with 2 compressor nodes

PCFG, and creating unit productions (with initial random probabilities) that rewrite from every slot-nonterminal to every other non-terminal (that is not a slot) or terminal.

To counter this objection, let me repeat the artificial language learning example of section 4.6.1, and assume that an HPN network with two compressor nodes, as illustrated in Figure 9.1, is trained on the following 3 sentences

(9.1) Mary walks.

(9.2) John walks.

(9.3) Sue likes John.

I argue that HPN can in principle generalize this to

(9.4) Sue likes Mary.

whereas the inside-outside algorithm cannot.

As was shown in the example of section 5.4.1, HPN produces the generalization by virtue of the topology: after the first 2 sentences have been analyzed by the compressor node labeled $X1$, *John* is moved closer to *Mary* in the topology (as a result of so-called 'contamination' via the slot S1). Subsequently, after analyzing sentence 3 with compressor node $X2$, slot S5 is moved closer to *John*, hence also closer to *Mary*.[1] Thus, in sentence 4 S5 is already expecting *Mary*. (Note that order is important.) The crucial feature in this example is *transitivity*: when two units/categories occur in similar contexts, part of the extension (i.e., vector representation in substitution space) of the first unit is transitively transferred to the second unit.

EM, on the other hand, may have learned that S1 rewrites to either *John* or *Mary*. However, a naive application of the inside-outside algorithm will not infer from this, plus $S5 \rightarrow John$ that S5 has a higher probability to rewrite to *Mary*, because it sees no relation between *John* and *Mary*, or between S1 and S5, as in a

[1] Alternatively, a Kohonen, or SOM-like learning algorithm can be implemented that together with the winning node also adjusts its topological neighbors within a predefined neighborhood to some extent.

PCFG S1 and S5 have independent rewrite probability distributions. Only with considerable effort one could construct a non-trivial prior that would teach EM how to generalize, or alternatively one could use cross-validation techniques.[2]

One may of course still question whether it is desirable that a learning system has these transitive properties. Although there is little empirical data to back this up, the generalization of the example corresponds to our intuition that the human language faculty exhibits *strong systematicity* [Hadley, 1994], as discussed in section 4.5.1: the capacity to generalize a category or word to novel syntactic positions where it has not been encountered before.

The transitive properties of metric learning impart upon HPN a non-linear dynamics, that would be very difficult to express as a prior within a Bayesian learning framework. In general, the approach to category learning advocated by HPN differs from standard machine learning techniques for classification, because in HPN syntactic categories derive their extensional meaning from their distributional role with respect other categories, rather than in isolation; HPN thus represents a holistic approach to category learning, and an inferential semantics, in which only the distances (and similarities) within the network topology are meaningful.

9.4.2 HPN versus Bayesian Model Merging (BMM)

Bayesian Model Merging (BMM) [Stolcke and Omohundro, 1994, Stolcke, 1994] was discussed in section 3.2.2 as a non-parametric alternative to the inside-outside algorithm, since it learns not only the parameters (probabilities), but also the structure (rules) of the grammar. In contrast to the inside-outside algorithm BMM can make strongly systematic generalizations. For instance, if in the above example *John* and *Mary* occur in the same (left-hand side) context, they can be *merged* into a single category. However, in BMM merging is a discrete, all-or-nothing operation. From the moment that *John* and *Mary* are merged they are indistinguishable. BMM is not able to express gradedness or similarity between categories, for instance between *adjectives* and *adverbs*, neither can it represent that similar categories share the same behavior in certain contexts: two categories are either identical or distinct.

As mentioned before, BMM clearly does not satisfy the connectionist constraint of locality; the search for an optimal grammar in BMM follows a global criterion (MDL), and after a merge the merged categories are substituted system-wide in the entire corpus.

[2]Thanks to Markos Mylonakis for this elaboration.

9.4.3 HPN versus U-DOP

U-DOP, or unsupervised Data Oriented Parsing [e.g., Bod, 2006b, 2007] is an extension of DOP to the domain of unsupervised parsing. As usual in DOP, the general approach is to allow all tree fragments to parse the sentence and let statistics decide. Whereas DOP is computationally quite expensive as a result of the extremely large number of fragments that can be extracted from a corpus (Sangati and Zuidema [2011] calculated that approximately 10^{48} (!) different fragments can be extracted from the Wall Street Journal), in U-DOP the combinatorial explosion is even worse, because for every sentence first all possible binary parse trees are generated, and subsequently from these all possible subtrees are extracted. In practice though, by means of Goodman's reduction [Goodman, 1996], the number of rules is kept manageable, and to date U-DOP is the most successful system for fully unsupervised bracketing, with an F-score of $78,5\%$ on the Penn Tree Bank (for sentences up to length 10).

Interestingly, the idea of learning a grammar from minimal difference pairs, by parsing with the shortest derivation, can in principle also be adopted for U-DOP, provided it parses the sentences of a corpus in an incremental fashion, as a way to keep the number of fragments manageable. As a cognitive model of grammar acquisition U-DOP is less plausible, because, like I-O and BMM, U-DOP assumes discrete categories, and a global search operation. Further, in the standard version of U-DOP it is assumed that all syntactic categories have the same label, hence the system is designed to learning brackets but not labels.

9.5 Future work

This thesis presented innovative ideas in many fields, such as cognitive linguistics, statistical parsing, neural network research and language acquisition, of which I have only started to exploit the possibilities. Below I present only a small selection of possible directions for future research.

Language model for speech recognition

While it is known that PCFG-models perform poorly as language models for speech recognition, probabilistic left corner models are often used for this task because they parse and condition the probabilities from left to right. The left corner parser of van Uytsel et al. [2001] has been evaluated for perplexity on several bench mark tests, with results that are competitive with NGram models. The non-episodic LCSG parser that I have implemented for this thesis is except for minor details equivalent to the van Uytsel et al. [2001] parser, hence is expected to perform at a similar level. (However, unlike van Uytsel et al. [2001] I have implemented smoothing.) It will be interesting to know how the episodic LCSG does as a language model with respect to the non-episodic LCSG.

What parsing strategy is employed by the human language processing system?

Do humans syntactically process sentences in a top-down, left-corner or other manner? To date, variations of the PCFG (top-down) are the state-of-the-art in syntactic parsing, but some of the best performing parsers employ features that are borrowed from left corner models [e.g. Charniak, 2000]. The episodic parsing framework allows for varying only the processing strategy between top-down, left-corner and bottom-up, while leaving all other parameters intact (in all cases the model's decisions are based on the entire derivational history of the training corpus), hence the framework offers an opportunity to make a fair comparison as to which processing strategy performs the best (with respect to F-scores or any other measure). This could answer an important question in cognitive linguistics research. Although I have performed a comparison between the episodic left corner and episodic top-down *reranker* (section 6.2.5), I have not done it for the episodic parser, because for that I would have to implement a standalone top-down episodic parser.

Episodic Markov Model

The concept of an episodic memory for sentence analyses, stored in the form of traces in syntactic nodes is not only applicable to context free grammars or left corner grammars, but also to regular grammars. It should not be too difficult to implement an episodic Markov Model, in which case episodic traces are stored in 'word nodes', and point to traces of previously processed sentences in other word nodes. I expect the properties of such a variable length Markov model to be comparable to those of the SRN, but much faster to train (in a single pass through the data). It will be interesting to evaluate the episodic Markov Model as a language model in a speech recognition task, and compare it to fixed length Markov Models, for instance a trigram model.

Are parse trees psychologically real?

Does the human mind represent sentences mentally as parse trees? This is the subject of an ongoing debate, as most connectionists reject the notion of hidden hierarchical sentence representations for lack of observable evidence. One can only try to address this fundamental question empirically, by designing behavioral and observable measures on which the predictions of a hierarchical language model that employs tree structures differ from those of a flat, non-hierarchical model. The fact that the chart parser implemented for the left corner shifting grammar (LCSG) is able to compute sentence and string probabilities (i.e., by summing over all parse trees that yield the same string) provides an excellent opportunity to compare it directly to non-hierarchical models, such as the SRN. A standard

way to assess the goodness of fit of a language model is by measuring the so-called perplexity of the corpus, that is the log likelihood of the data given the model. Thus, one might compare the likelihood, or perplexity of sentences (not parses) between the hierarchical left corner model (LCSG), or its episodic version on the one hand and the SRN or the flat episodic Markov Model (see the previous paragraph) on the other hand.

Another option, which has recently received much interest in the literature, is to relate the predictions of the model to *reading times*. According to *surprisal theory* [e.g., Hale, 2001, Levy, 2008] it may be assumed that reading times are linked to the 'surprisal' of a word, that is, how much the word is expected. The latter value can be estimated by any probabilistic language model, provided it can compute string probabilities (and provided lexical semantic confounds can be factored out). While several studies have investigated the effect of 'syntactic' (that is, unlexicalized) surprisal, as well as 'integration cost', on reading times [e.g., Demberg and Keller, 2008, Boston et al., 2008], Frank and Bod [2011] compared the predicted reading times, based on syntactic surprisal, between a PCFG and the SRN, and conclude that the human sentence processing system is insensitive to hierarchical structure. However, since they compared the SRN only to a PCFG model their results may have been biased, because in the SRN left-to-right processing is built into the system, whereas in the PCFG model the computed left-to-right transition probabilities do not fall out in any cognitively motivated way from the top-down probabilistic model. It would therefore be interesting to see whether their results still hold if the SRN is compared to the probabilistic LCSG, or the episodic LCSG.

Yet another option for assessing the psychological reality of trees is to investigate the effect of syntactic priming on reading times. For instance, Sturt et al. [2010] have shown that if two noun phrases in a sentence share the same structure processing is facilitated in the second one.

Modeling language acquisition

The potential of HPN as a tool for language acquisition research was already mentioned in section 9.2.1. An obvious choice for an experiment is to try to replicate the results of [Borensztajn et al., 2009a], in which we found evidence that 'children's grammars grow more abstract with age', with the episodic-HPN model. In the cited work, we used tools from computational linguistics, specifically the push-'n-pull estimator for Data Oriented Parsing [Zuidema, 2007] to do the statistical analysis. However, the analysis was based on sentences from the CHILDES corpus [MacWhinney, 2000] that were manually labeled with adult syntactic labels. Using episodic-HPN, we have a fully unsupervised language model, that can induce clusters of syntactic categories, and episodic constructions based on sentences without any label annotation. Moreover, the episodic fragments that episodic-HPN finds are fragments from a left corner derivation,

which may form an interesting comparison with the top-down DOP-fragments found by push-'n-pull.

It would be extremely interesting if it were possible, using episodic-HPN, to inspect snapshots of the syntactic topology ('substitution space') at different stages in children's linguistic development (for example Adam, Eve and Sarah from the CHILDES database), and particularly to track the changes and self-organization process of their grammars. A preliminary experiment showing Eve's topology was performed in section 5.5.3, but this was done with the 'semantic' HPN model.

Unsupervised grammar induction with episodic-HPN

In principle episodic-HPN should be suitable for the task of unsupervised grammar induction from realistic and real-sized corpora, yet in practice there are still many technical obstacles to overcome before this can be realized. What is interesting about episodic-HPN is that is finds brackets and labels (that is, clusters) simultaneously, unlike many unsupervised induction systems that focus only on bracket induction [e.g., Bod, 2006a, Klein and Manning, 2002]. This may be an advantage if one assumes that the induction of brackets depends on the induced labels and vice versa, and thus brackets and labels can bootstrap each other [e.g., Borensztajn, 2006b, Stolcke and Omohundro, 1994].

On the other hand, it should be computationally much less expensive to invoke episodic-HPN as a system for unsupervised labeling, with given brackets. It will be interesting to compare the performance of episodic-HPN on this task with that of the Bayesian Model Merging approach [e.g., Borensztajn and Zuidema, 2007, Reichart and Rappoport, 1992].

Syntactic priming and recency effects

With only slight modifications the episodic grammar can be made sensitive to recency effects in sentence processing. Since the sequential numbers of the exemplars are encoded in the traces in chronological order, it will be easy to implement a forgetting or decay function that depends on the age of the trace. This allows investigating the effects of forgetting unused episodes, while for example entrenching episodes or episodic fragments that are frequently used. Similarly, the episodic framework lends itself to an investigation of syntactic priming effects, that is the influence of the use of specific syntactic patterns across the sentence boundary in dialog.

Learning with a topology

There is an entire research field awaiting further exploration regarding the study of the dynamics of topology learning and self-organization in grammar acquisition, and its relation to empirical data. To gain a better understanding of

the subject one might start by analyzing cases for which the learning dynamics of topological self-organization in HPN makes different predictions than standard machine learning techniques, or Bayesian approaches to learning. I have suggested in section 9.4.1 that topology learning predicts certain generalizations that are not predicted by the Inside-Outside algorithm from the same data, yet the question whether such generalizations are a better fit to the empirical data on child language acquisition has yet been to be answered. To date not much is known from the language acquisition literature about the way children generalize their constructions, nor whether this points to some topological organization of their grammars.

Improvements and extensions of the episodic grammar framework

There remain many engineering problems to be solved for the episodic framework (as well as for the HPN framework), as well as optional extensions that are worth trying out. For instance, by leaving out the shift transitions in the shortest derivation parser it can only approximate the shortest derivations, and performance is likely not optimal. On the other hand, if the shift counts are included one may obtain fragments from the shortest left corner derivation containing words, which are probably more informative than the wordless fragments obtained with the current method.

Another boost to the performance of the episodic shortest derivation parser is expected if the model can take discontiguous fragments into account. In section 7.3.2 I hinted at how it can be done, but it has not yet been implemented.

A probabilistic episodic chart parser may be an interesting direction, as it implements the principle of an episodic spreading activation parser, where episodic probabilities are computed through spreading activations (i,e, Common History) between the traces in the states of the chart parser.

Neuro-biological predictions

This thesis makes numerous predictions about, among others, the neural correlates of syntactic categories, the organization of semantic and episodic memory and the functional connectivity of the cortex. For some of these predictions today's technology may not yet be sufficiently sophisticated to allow their empirical verification.

To start with, the hypothesis for a neural theory of syntax (section 2.8) predicts that dedicated cell assemblies in the cortex are able to represent graded syntactic category membership; it is hypothesized that the 'syntactic' cell assemblies, presumably encoded in so-called 'name fields' of cortical columns, communicate with each other through serial dynamic binding: they pass 'tags' to a 'switchboard' (assumed to be implemented by the hippocampus), where they are redirected to other, topologically neighboring cell assemblies. This mode

of communication by the cortex guarantees context invariant processing. The paradigmatic relations of a syntax (i.e., substitutibility relations between cell assemblies) are encoded in the 'substitution space', which topologically organizes the cell assemblies according to representations that express a neural assembly's graded syntactic category membership. Note that the theory does not predict the syntactic topology to be spatially laid out along the cortical surface, but rather it predicts that topological addresses are stored in 'name fields', and transmitted through spiking patterns.

The theory further predicts that syntactic assemblies (cortical columns) have the ability to temporally integrate ordered sequences of inputs, hence function as compressor node; a possible neural implementation was proposed in section 5.7. They must possess local memories, where first pointers to the addresses of bound assemblies are temporarily stored, and after a successfull parse the pointers are stored as persistent memory traces. A working memory, or *stack*, is implemented as a set of active pointers distributed over assemblies that participate in a parse.

Appendix A

Two case studies on the systematicity of the SRN

The systematicity issue of the SRN is addressed in a series of influential papers by Jeffrey Elman [Elman, 1990, 1991, 1993], in which he famously formulates the connectionist claims about the 'implicit' nature of grammatical knowledge in distributed networks. These publications have had an enormous impact, as they succeeded to convince many connectionists that distributed neural networks can generalize systematically without any need for explicit representation of syntactic categories or constituents. Such ideas have contributed for a large part to the notion that emergent rule-like behavior can be explained without explicit representation of rules or categories, which has become the standard connectionist response to Fodor and Pylyshyn [1988] in the systematicity debate.

Yet, in Chapter 4 I have argued on theoretical grounds that the SRN does not satisfy either the context invariance criterion or the recursive systematicity criterion. In the following two case studies I want to critically assess Elman's claims about the, in my opinion, problematic idea of 'implicit representation' of categories.

A.1 Case study 1: Elman [1990]

Elman [1990] describes an experiment where an SRN is trained on $10,000$ sentences of 2 or 3 words generated at random from 16 templates. The templates used 29 different lexical items selected from 6 different noun classes (for instance NOUN-HUM and NOUN-ANIM) and 6 verb classes, among which intransitive and transitive verbs. The task of the network was to learn to predict the next word in the sentence (the 'word prediction task'), and this was trained using error back propagation with the correct word as the target. For the test sequences the network cannot be expected to predict the exact target word, since all words of a category are equally probable. Therefore, the output was evaluated against a

target vector consisting of the expected frequencies of all 29 words, as derived from a second order Markov model trained on the same data. After training for 100 epochs the network succeeded to predict the next word class with a very low root mean squared error.

To investigate how the network accomplished this feat, Elman analyzed the hidden layer representations developed by the network. The activity of the hidden layer (150 units) was measured after every word in the training set, and for every unique word an average activation vector was obtained over all previous contexts of the word. Subsequently a hierarchical cluster analysis (HCA) was performed on these averaged vectors. This resulted in word clusters that resembled the original categories from the artificial grammar to a large degree. Elman concluded from these results that the SRN had discovered and internalized *implicit* knowledge about the major word categories. This would suggest that the SRN exhibits strong systematicity in the sense of [Hadley, 1994].

Implicit knowledge is an empty concept

What is the 'implicit knowledge' of word categories that Elman [1990] is referring to? It makes sense that the term 'category' in the context of a particular model is reserved to denote a set of entities which share the same behavior under a certain operation of the model. If one speaks of the 'categories' of the SRN one would expect this notion to refer to a subset of its states for which the future output of the SRN is identical. Only with this intended interpretation of the term 'category' in mind can one have a productive discussion on the generalization or systematicity of the SRN.

Yet, in the hierarchical cluster analysis of [Elman, 1990], the so-called categories that are constructed by averaging over contexts are not causative of any behavior of the SRN, and therefore are not genuine categories in the above sense. Members of Elman's categories (different contexts of the same word) do not produce the same output behavior. For instance, the word *boy* at the beginning of a sentence does not predict the same next word as in the context *dog chases boy*. Since the 'implicit categories' never actually appear in the SRN, but are the result of a post-hoc analysis, it is not justified to appeal to them as the cause behind systematic behavior of the SRN, nor is it warranted to ascribe these constructs to the SRN's 'implicit knowledge'. In fact the knowledge implicit in the average context vectors is not internally encoded in any useful way by the SRN, but it is merely a redescription of the input data.

To verify the latter claim I have performed a control experiment, in which I applied HCA directly to the trigram model that produced the target vectors used to train the SRN (thus, using the same data). Strikingly, this analysis revealed an almost identical hierarchy of categories as those found in the hidden layer of the Elman network, in a single pass through the data (compare Figures A.1 and A.2). This strongly suggests that the clusters found by the HCA analysis do not reflect

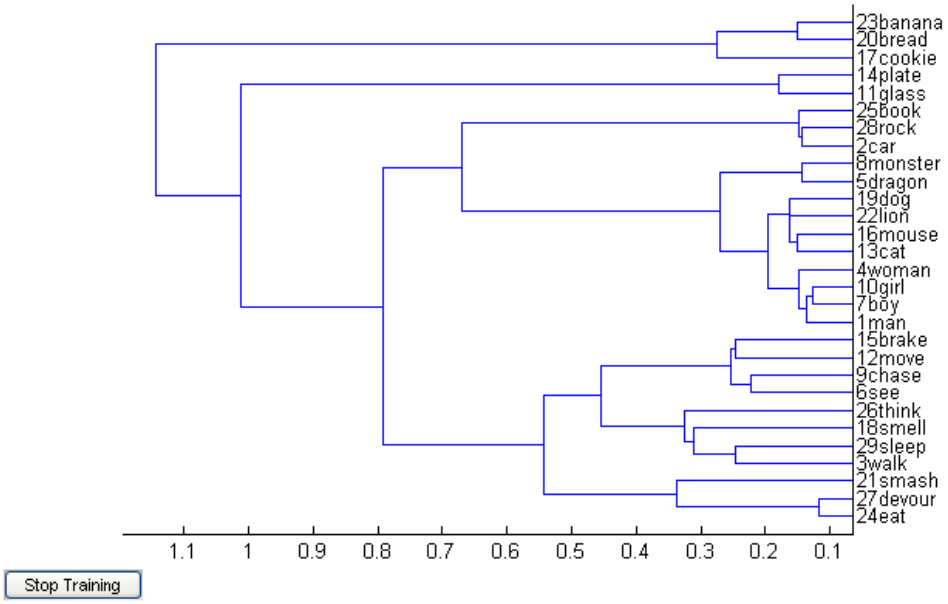


Figure A.1: Hierarchical cluster analysis of hidden layer of the SRN

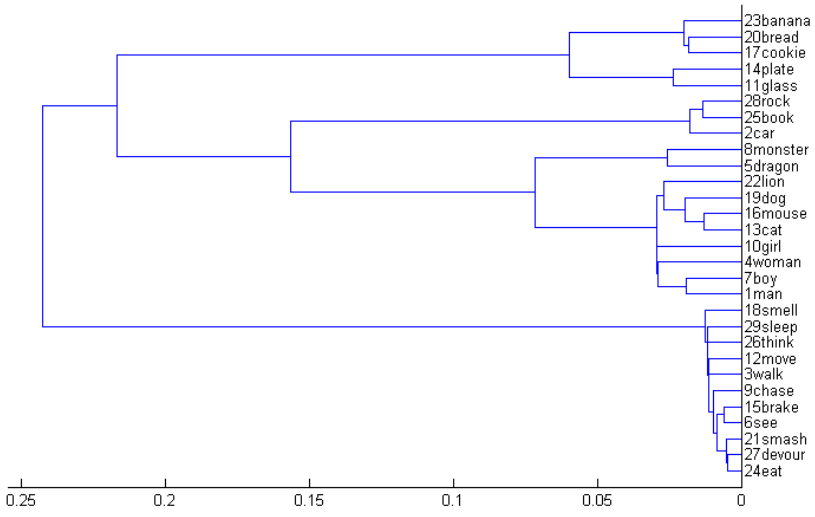


Figure A.2: Hierarchical cluster analysis of trigrams, for the same data as used in Figure A.1

how the network represents categories, but that they are an intrinsic property of the input distribution.[1]

A.2 Case study 2: Elman [1991]

In a second experiment Elman [1991] explored whether and how the SRN represents complex structural relationships such as constituency. The goal was "to see if the network could infer the constituency structure from stimuli in which underlying hierarchical relationships were hidden, and represent the compositional relationships in such a manner as to support structure-sensitive operations" [Elman, 1991, p.199].

Training sentences were generated from a recursive artificial context free grammar that included categories for singular and plural nouns (e.g., *boy(s)*, *girl(s)*), transitive verbs (e.g. *feed*, *chase*) and intransitive verbs (e.g. *walk*, *live*), both singular and plural, and relative clauses (*who*). Number agreement was enforced between subject and verb, and between subject and relative clause. The SRN was tested on the word prediction task, as in [Elman, 1990], but since relative clauses are embedded in the sentence, number agreement and other dependencies must be maintained over longer distances.

While the SRN succeeded on this task at least in some trials, the question that Elman asked is how grammatical knowledge about constituent structure, argument structure, grammatical category and number agreement are systematically encoded in the network, and whether this knowledge allows for systematic inferences and generalization to novel examples? To this end Elman applied principle component analysis (PCA) to analyze the activation space of the hidden units, and plotted the dimensions of highest variance. With regard to encoding the depth of recursive embedding, Figure A.3 shows the network trajectory in state space for

(A.1) boy chases boy who chases boy who chases boy .

From Figure A.3 it is apparent that embeddings are distinguished from each other, and from the main clause, by their displacement. Moreover, from inspection of the graphs of the other sentences it seems that the solution exhibits a systematic pattern. This prompted Elman to conclude that "displacement provides a systematic way for the network to encode the depth of embedding" [Elman, 1991, p.111]. A more ambitious proposition is also implied [Elman, 1991, p.110], namely that the grammatical knowledge of the SRN is not encoded in static representations, but in the temporal dynamics of the network; in particular, Elman claims that

[1] Elman [1990] remarks in a footnote (6) that comparable results as for a trained SRN can be obtained by an SRN even without learning (and using uniform weights). Thus, apparently the author is aware of this problem, but nevertheless draws no conclusions from it.

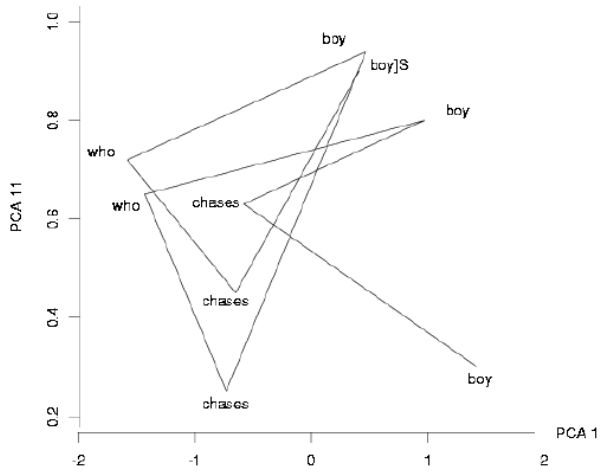


Figure A.3: Trajectory through state space of the sentence *boy chases boy who chases boy who chases boy.* Reproduced from [Elman, 1991]

the knowledge of the network is encoded in (constraints on possible) temporal trajectories through state space.

Knowledge about displacement is attributed post-hoc

Elman's claims about the 'implicit knowledge' of the SRN of constituent structure are prone to the same critique that applies for his claims about 'implicit knowledge' of word categories (section A.1): the knowledge of 'displacement' is not internally encoded in a form that the SRN can exploit or access, but rather attributed post-hoc.

In principle the only available information to the SRN to base its decisions upon at time t is the input activation plus the state of the hidden units at time $t-1$. To make use of displacement information the SRN would need to perform a meta-analysis that compares the state of the hidden units at different time steps. Perhaps, it would be possible to build into the network a mechanism designed to extract differential information, for instance by adding units that act as 'difference detectors' between hidden states at different times. However, this will not solve the problem of long distance dependencies, because crucially one would need to compare hidden states that correspond to matching recursive levels, which are not at fixed distances (that is, temporal intervals) in the sentence. Hence, to decide which earlier hidden state to compare to which, the SRN would need to have access to prior knowledge about the level of embedding (i.e., it would need a stack), but this is precisely the task that the SRN was supposed to solve. In sum, the claim that displacement systematically encodes depth of embedding is unwarranted, and so is the broader claim that constituency and recursion are 'implicitly' encoded in the temporal dynamics of the network.

While it could still be possible that, despite Elman's apparently flawed analysis, the SRN does find a systematic solution for the structure encoding problem, there are fundamental reasons to believe that this is not the case. These are discussed in the next section.

A.2.1 Fundamental reason why an SRN cannot generalize context free languages: analogy

In this section I will argue that a solution for encoding recursion of the kind proposed by Elman [1991], in which constituents of different recursive levels are separated by their position (and predictive behavior) in state space, cannot *in principle* be generalized by the SRN in a systematic (i.e., non-arbitrary) way, but only through coincidence.

For convenience I will use the example of the context free language $a^n b^n$. Rodriguez et al. [1999] claim that the SRN can *learn* to generalize this language from a finite set of examples with $1 \leq n \leq 11$ to $n = 16$ (hence, the title of the paper is *A recurrent network that learns to count*), while using displacement to distinguish different levels of recursion, like the solution in [Elman, 1991].

To avoid any confusion let me reiterate that the fact that the SRN can *represent* a solution for unbounded recursion is beyond doubt (see also section 4.4). In fact, Rodriguez et al. [1999] give an example of a configuration of weights and initial hidden unit activations, such that a SRN with two hidden units, two context units and two output units can recognize arbitrarily long strings of the form $a^n b^n$, where a is represented as $(1, 0)$ and b as $(0, 1)$.[2]

However, the question of interest is not about the ability of the SRN to *represent* a context free language, but about its ability to *learn* it by generalizing from the training data. Suppose the SRN has been trained on '*a b*', '*a a b b*' and '*a a a b b b*'. I argue that the SRN will not find the solution proposed by Rodriguez et al. [1999] (or a similar solution) by means of systematic generalization.

[2]The weights of the SRN are chosen such that one hidden unit (X_1) counts the a inputs by increasing its activation monotonically, and the other hidden unit (X_2) counts the b units. A change of input between a and b turns the counting on or off. For example, given input units $\mathbf{I_t}$, hidden units $\mathbf{X_t}$ with initial value $\mathbf{X_0} = (0, 0)$, and given input sequence *aaabbb*; if one defines the weights and activation function $\mathcal{F}$ as follows

$$\mathcal{F}(net) = \begin{cases} 1 & net \geq 1 \\ net & 0 < net < 1 \\ 0 & net \leq 1 \end{cases}$$

$$\mathbf{X_t} = \mathcal{F}\left(\begin{bmatrix} 0.5 & 0 \\ 2.0 & 2.0 \end{bmatrix} \cdot \mathbf{X_{t-1}} + \begin{bmatrix} 0.5 & -5 \\ -5 & -1 \end{bmatrix} \cdot \mathbf{I_t} \right)$$

then the activations of the hidden units (the state space) follows a monotonically increasing and then decreasing sequence of values: $(0.5, 0)$, $(0.75, 0)$, $(0.875, 0)$, $(0, 0.75)$, $(0, 0.5)$, $(0, 0)$. For arbitrarily long strings $a^n b^n$ this series converges to the limit points (1,0) and (0,1).

Starting point of the argument is the observation, made in section 4.5.3, that one should be able to show that the generalization is *causally inferred* from some similarities in the training data with respect to the structure bias of the SRN. The first thing to realize in this respect is that there is nothing in the superficial similarity between the hidden unit activations induced by '*a b*', '*a a b b*' and '*a a a b b b*' that will cause the SRN to generalize correctly and predict the final *b* given '*a a a a b b*'. In fact, the train instance that produces the most similar activations in hidden unit space to '*a a a a b b b*' is '*a a a b b b*', yet this string predicts the end-of-sentence marker instead of *b*.

We may conclude that generalization on the basis of superficial similarity of distributed patterns does not produce *recursive systematicity*, hence does not offer the desired productivity of language. Yet, superficial similarity (or alternatively, interpolation within the training space) is the sole basis for generalization for an SRN that learns with error back propagation (EBP will cause superficially similar states to produce similar output).

As said, in case of recursive systematicity the causal inference for generalization is not based on superficial similarity, but on analogy: the similarity of relations. To see the analogy between '*a b*', '*a a b b*' and '*a a a b b b*' the system must discover that the operations that transform '*a b*' into '*a a b b*', and '*a a b b*' into '*a a a b b b*' are similar.[3] Suppose the SRN was able to discover such a systematic operation (i.e., a transition in state space that can be reused). Then, were the systematic solution applied to process a novel, previously unseen sentence, it would require the SRN to visit one particular hidden state at least twice. In other words, the solution would have to involve loops between states in hidden unit space.

Now, the solutions for encoding recursion proposed both by [Rodriguez et al., 1999] and by [Elman, 1991] crucially make use of the fact that constituents at every different level of recursion are separated by their position in state space (i.e., their different predictions are based on displacement). This contradicts our earlier conclusion that any solution that is reached by generalization from analogy must have loops, hence the solution of [Rodriguez et al., 1999] is not produced by generalization from analogy, but can only be reached through coincidence.

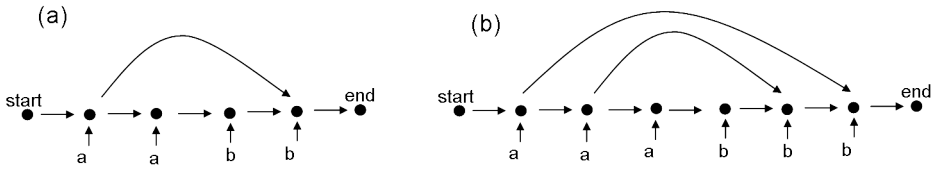


Figure A.4: FSA representing $a^n b^n$ for $n = 2$ (left) and $n = 3$ (right)

[3]See section 8.2.3 for a discussion of how a learning algorithm based on a CFG bias, such as the BMM model, discovers analogies in the training data.

In fact, the solution for learning recursive structure proposed by Rodriguez [1999] and Elman [1991] is comparable to the trivial solution of the FSA, given in section 4.6.1, which is also based on duplication of states for higher recursive levels (see Figure A.4). Clearly, the right FSA (b) in Figure A.4 cannot be inferred from the left FSA (a) by means of the discovery of analogies in the training data, because that would imply that a state is visited more than once, but then Figure A.4 (b) would need to have less states than Figure A.4 (a). The fact that the SRN has an infinitely large (continuous) state space does not give it any advantage in generalizing from a finite number of examples over the FSA, because the available strategies are the same in both cases and depend only on the structure bias: the ability to express analogies or transformations.

Appendix B

HPN implementation issues

B.1 A simple implementation of a deterministic and serial left corner parser for HPN

As explained in section 5.3.1, HPN implements a left corner parsing (LCP) strategy to search for a syntactic analysis of a sentence. The HPN implementation of a serial LC parser deviates in some minor details from the standard symbolic implementation of a LC parser (see also section 3.1.3). While as usual a derivation of a sentence involves *shift*, *attach* and *project* operations, in the description of the parse process I have tried to consistently use the terminology of HPN.

Table B.1 includes pseudo-code for a simple deterministic and serial HPN left corner parser. The `Main` method calls the `bottomUpProcess` method, which starts a derivation by shifting to an input unit corresponding to the first word of the sentence, and projects it to a compressor node. Most of the work is done in two methods: the `bottomUpProcess` method, in which the shift, attach and project operations are included, and the `topDownProcess` method, in which an HPN production (of a compressor node) iterates through its goal slots (top-down prediction). In the `bottomUpProcess` a partial parse structure is projected upward from a word to the left slot of a compressor node, and eventually has to attach to a free non-left slot of a compressor node. If there is no `match` between the projected structure and a free slot the parser has to backtrack: it returns the call and attempts a projection to a different compressor node. The `timestep` (i.e., the word position) and a pointer to nodes that are bound to the slots must be passed between the methods. The latter is implicitly handled by the Java call stack (local node memories are assumed). The `timestep` variable corresponds to the index number of a *node state*, as will be explained in section 5.3.2. The `timestep` is incremented by 1 upon shift, and propagated through the bindings. In the pseudo-code `bottomupTopVector` denotes the vector representation of the root of a compressor node or input node that is derived bottom-up, and `activeSlotVector` denotes the vector of the first free slot of a compressor node.

```
method Main
    bottomUpProcess(timestep, null, null)


method bottomUpProcess(timestep, activeSlotVector, completedCompressorNode)
    % left corner is either the root of compressorNode or next word in the sentence
    if (completedCompressorNode exists)
        bottomupTopVector= completedCompressorNode
    else % shift:  set bottomupTopVector to the input node representation
    % corresponding to the next word
        timestep = timestep +1
        bottomupTopVector = inputNodes(wordsOfSentence(timestep))

    % attach:  match input with open slot from active production
    if (activeSlotVector exists)
        if (innerProduct(bottomupTopVector, activeSlotVector)>0)
            return (match, timestep)

    % project:  loop over all compressor nodes; try to match bottomupTopVector to
    % the left slot of any of the compressor nodes
    for (each compressorNode in compressorNodes)
        if (innerProduct(bottomupTopVector, leftSlotOfCompressorNode)>0)
            match = compressorNode.topDownProcess(timestep, activeSlotVector)
        if (match) return (match, timestep)
    % failure
    return (failure, timestep)


method compressorNode.topDownProcess(timestep, goalSlot)
    % loop over non-left slots of the current compressor node
    for (activeSlotVector = second slot, ..., last slot)
        match = bottomUpProcess(timestep, activeSlotVector, null)
        % (there can be a match if either the next word in the sentence matches directly
        % to the activeSlot, or a projected compressorNode matches the activeSlot)
        if (match) return (match, timestep)
    end for % production complete

    % goalReachedCheck
    if (goalSlot exists)
        if (innerProduct(this.RootVector, goalslot)>0) return (match, timestep)
        else % call from Main:  check whether all words of the sentence have been processed
            if (timestep==sentence.size()) return (match, timestep)

    % no match with goalSlot, and no end of sentence, then
    % try projecting up from the current compressor node (left-branching)
    match = bottomUpProcess(timestep, goalSlot, this.RootVector)
    if (match) return (match, timestep)
    else return (failure, timestep)
```

Table B.1: Pseudocode for a deterministic and serial left corner parser algorithm in HPN

B.2 Conversion procedure from (P)CFG to (P)HPN

In section 5.2 I argued that context free grammars are subsumed in the HPN formalism. To prove this, in this section I will sketch 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, the nodes and slots are represented as vectors with respect to an orthogonal basis of slots. One 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	→	NP VP (1.0)
NP	→	PropN (0.2) ‖ N (0.5) ‖ N RC (0.3)
VP	→	VI (0.4) ‖ VT NP (0.6)
RC	→	WHO NP VT (0.1) ‖ WHO VP (0.9)
VI	→	walks (0.5) ‖ lives (0.5)
VT	→	chases (0.8) ‖ feeds (0.2)
N	→	boy (0.6) ‖ girl (0.4)
PropN	→	John (0.5) ‖ Mary (0.5)
WHO	→	who (1.0)

Table B.2: Toy probabilistic context-free grammar with relative clauses (Adapted from [Elman, 1991]). 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 5.4 (using informal notation, and with unary productions added for clarity), for the CFG grammar with recursive relative clauses given in Table B.2.

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*.

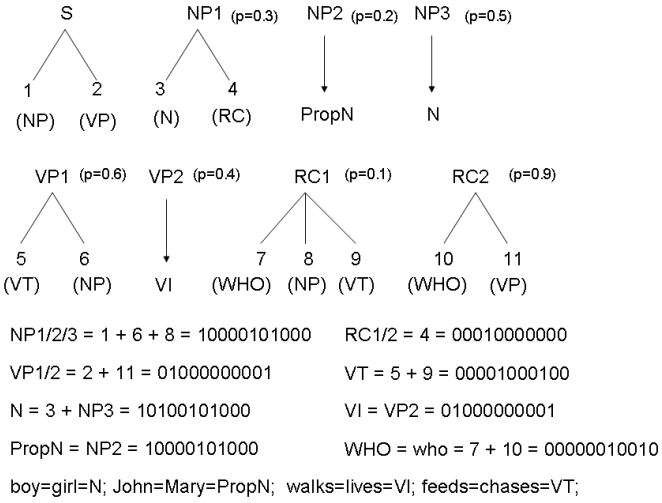


Figure B.1: Conversion procedure from CFG to HPN. The slot indices reflect their only non-zero component.

It is easy to modify the conversion procedure such that it converts a probabilistic context free grammar (PCFG) into a probabilistic version of 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 B.3 gives the 'probabilistic' node representations (assuming

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 B.3: Node representations for probabilistic HPN.

the HPN productions of Figure B.1). It can be shown that, using this conversion procedure, for any parse of a sentence generated by the artificial PCFG grammar there exists a parse by the corresponding HPN grammar with the same branching structure and the same probability as assigned by the PCFG.

Bibliography

M. Bansal and D. Klein. The surprising variance in shortest-derivation parsing. In *Proceedings of the 49nd Annual Meeting of Association for Computational Linguistics (ACL 2011)*, pages 720–725, Morristown, NJ, 2011. Association for Computational Linguistics. [152, 158, 159, 161]

O. Barak and M. Tsodyks. Persistent activity in neural networks with dynamic synapses. *PLOS computational biology*, 3(2):e35, 2007. [25, 116]

F. Battaglia and C. M. A. Pennartz. The construction of semantic memory: grammar based representations learned from relational episodic information. *Frontiers in computational neuroscience*, 5(36):1–22, 2011. [31, 176]

F. Battaglia, K. Benchenane, K. Sirota, C. M. A. Pennartz, and S. Wiener. The hippocampus: hub of brain network communication for memory. *Trends in Cognitive Sciences*, 2011. [32]

W. Bechtel and A. Abrahamsen. *Connectionism and the mind: Parallel processing, dynamics, and evolution in networks. Second Edition.* Basil Blackwell, Oxford, UK, 2002. [67]

B. K. Bergen and N. C. Chang. Embodied construction grammar in simulation-based language understanding. In J. Östman and M. Fried, editors, *Construction grammars: cognitive grounding and theoretical extensions*, pages 147–190. 2005. Technical Report 02-004. [182]

C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1996. [65]

E. Black, F. Jelinek, J. Lafferty, D. Magerman, and R. Mercer. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 31–37, 1993. [136]

R. Bod. *Beyond Grammar: An experience-based theory of language*. CSLI Publications, Stanford, CA, 1998. [5, 30, 50, 136]

R. Bod. Parsing with the shortest derivation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 69–75, -, 2000. Association for Computational Linguistics. [135, 161, 168]

R. Bod. An efficient implementation of a new DOP model. In *Proceedings EACL'03*, pages 19–26. 2003. [7, 49, 158, 161]

R. Bod. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics, Sydney*, pages 865–872, 2006a. [191]

R. Bod. Unsupervised parsing with U-DOP. In *Proceedings of the 10th International Conference on Computational Natural Language Learning (CONLL-X)*, pages 85–92, 2006b. [188]

R. Bod. Is the end of supervised parsing in sight? In *Proceedings of the 45nd Annual Meeting of Association for Computational Linguistics (ACL 2007)*, volume 45, page 400, Morristown, NJ, 2007. Association for Computational Linguistics. [55, 188]

G. Borensztajn. Luc steels, the Talking Heads experiment and cognitive philosophy - a tutorial accompanying the presentation in current issues - part i. available at http://staff.science.uva.nl/g̃ideon/, 2006a. [37]

G. Borensztajn. Automatic discovery of constructions in children's speech, masters thesis, 2006b. [16, 191]

G. Borensztajn and W. Zuidema. Bayesian model merging for unsupervised constituent labeling and grammar induction. Technical report, 2007. [191]

G. Borensztajn and W. Zuidema. Episodic grammar: a computational model of the interaction between episodic and semantic memory in language processing. In *Proceedings of the 33th Annual Conference of the Cognitive Science Society*, 2011. [9]

G. Borensztajn, W. Zuidema, and R. Bod. Children's grammars grow more abstract with age. evidence from an automatic procedure for identifying the productive units of language. *Topics*, 1(1):175–188, 2009a. [5, 7, 51, 190]

G. Borensztajn, W. Zuidema, and R. Bod. The hierarchical prediction network: towards a neural theory of grammar acquisition. In *Proceedings of the 31th annual conference of the cognitive science society*, 2009b. [8]

M. Boston, J. Hale, R. Kliegl, J. Patil, and S. Vasisht. Parsing costs as predictors of reading difficulty: An evaluation using the potsdam sentence corpus. *Journal of Eye Movement Research*, 2(1):1–12, 2008. [190]

P. Brakel and S. Frank. Strong systematicity in sentence processing by simple recurrent networks. In *Proceedings of the 31th annual conference of the cognitive science society*, 2009. [75]

R. W. Brown. *A first language: The early stages*. Harvard University Press, Cambridge, MA, 1973. [51, 108]

M. Bunsey and H. Eichenbaum. Conservation of hippocampal memory function in rats and humans. *Nature*, 379(6562):255–257, 1996. [34]

N. C. Chang. *Learning Grammatical Constructions*. PhD thesis, University of California, Berkeley, 2001. Thesis proposal. [37]

E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, 2000. [49, 131, 189]

N. Chomsky. *Syntactic Structures*. Mouton & Co, The Hague, 1957. [42, 43]

N. Chomsky. *Language and mind*. Harcourt, Brace and World, -, 1972. Extended edition. [66]

N. Chomsky. *Lectures on Government and Binding*. Foris Publications, Dordrecht, the Netherlands, 1981. [59]

N. Chomsky and G. A. Miller. Introduction to the formal analysis of natural languages. In D. Luce, X. Bush, and X. Galanter, editors, *Handbook of Mathematical Psychology, Vol. 2*, pages 269–321. 1963. [43]

M. H. Christiansen and N. Chater. Generalization and connectionist language learning. *Mind and Language*, 9(3):273–287, 1994. [75]

M. H. Christiansen and N. Chater. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23(2):157–205, 1999. [79, 82, 84, 180]

A. M. Collins and M. Quillian. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247, 1969. [27]

M. J. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, 1999. [136]

M. J. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589 – 637, 2003. [49]

N. Cowan. *Attention and memory: An integrated framework*. Oxford University Press, Oxford, UK, 1995. [116]

S. Crain. Language acquisition in the absence of experience. *Behavioral and Brain Sciences*, 14(4):597–650, 1991. [66]

S. Crain and R. Thornton. Acquisition of syntax and semantics. In M. Traxler and M. Gernsbacher, editors, *Handbook of Psycholinguistics*. Elsevier, Oxford, 2005. [52]

F. Crick and C. Koch. Consciousness and neuroscience. *Cerebral Cortex*, 8(2): 97–107, 1998. [11, 59]

W. Croft. *Syntactic categories and grammatical relations: The cognitive organization of information*. University of Chicago Press, Chicago, IL, 1991. [5]

W. Croft. *Radical construction grammar: syntactic theory in typological perspective*. Oxford University Press, Oxford, UK, 2001. [5, 23, 59, 60, 182]

V. Demberg and F. Keller. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210, 2008. [190]

J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970. [52, 104, 140]

S. Edelman. *Computing the mind. How the Mind Really Works*. Oxford University Press, Oxford, UK, 2008. [83]

H. Eichenbaum. Hippocampus: Cognitive processes and neural representations that underlie declarative memory. *Neuron*, 44(1):109–120, 2004. [32, 33, 34, 123, 138, 172, 174, 175]

H. Eichenbaum and N. J. Cohen. *From Conditioning to Conscious Recollection: Memory Systems of the Brain*. Oxford University Press, Oxford, UK, 2001. [177]

H. Eichenbaum and N. J. Fortin. The neurobiology of memory based predictions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1183–1191, 2009. [34, 177]

H. Eichenbaum, C. Stewart, and R. G. Morris. Hippocampal representation in spatial learning. *Journal of Neuroscience*, 10(11):3531–3542, 1990. [34]

H. Eichenbaum, P. Dudchenko, E. Wood, M. Shapiro, and H. Tanila. The hippocampus, memory, review and place cells: Is it spatial memory or a memory space? *Neuron*, 23:209–226, 1999. [33, 34]

J. L. Elman. Representation and structure in connectionist models. Technical report, University of California at San Diego, La Jolla, 1989. CRL 8903. [8, 73]

J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. [xi, 74, 84, 87, 195, 196, 197, 198]

J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2):195–225, 1991. [xi, 67, 74, 87, 100, 107, 195, 198, 199, 200, 201, 205]

J. L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993. [84, 195]

N. Evans and S. C. Levinson. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioral and Brain Sciences*, 32(5):429–448, 2009. [23]

D. Everett. Cultural constraints on grammar and cognition in pirahã. *Current Anthropology*, 46(4):621–646, 2005. [23]

D. J. Felleman and D. C. van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1(1):1–47, 1991. [12, 14]

C. J. Fillmore and P. Kay. The goals of construction grammar. Technical report, 1987. Berkeley Cognitive Science Program TR. [5]

J. D. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988. [3, 66, 67, 69, 74, 76, 90, 183, 195]

S. Frank and R. Bod. Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science*, 22(6):829, 2011. [190]

P. W. Frankland and B. Bontempi. The organization of recent and remote memories. *Nature Reviews Neuroscience*, 6(2):119–130, 2005. [175, 176]

G. Frege. Über sinn und bedeutung [on sense and reference]. *Zeitschrift für Philosophie und philosophische Kritik*, 1892. [58]

D. George and J. Hawkins. A hierarchical Bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the 2005 IEEE International Joint Conference on neural networks*, volume 3, pages 1812–1817, 2005. [2, 19, 182]

D. George and J. Hawkins. Towards a mathematical theory of cortical microcircuits. *PLOS computational biology*, 5(10):e1000532, 2009. [2, 20, 37, 182]

A. E. Goldberg. Constructions: a new theoretical approach to language. *Trends in Cognitive Sciences*, 7(5):219–224, 2003. [5, 36, 182]

A. E. Goldberg. *Constructions in Context*. Oxford University Press, Oxford, UK, 2006. [5, 30, 36, 182]

J. Goodman. Efficient algorithms for parsing the DOP model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 143–152. 1996. [159, 188]

R. F. Hadley. Systematicity in connectionist language learning. *Mind & Language*, 9(3):247–272, 1994. [viii, 8, 63, 74, 75, 80, 86, 187, 196]

P. Hagoort. On Broca, brain, and binding: a new framework. *Trends in Cognitive Sciences*, 9(9):416–423, 2005. [117]

J. Hale. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, volume 2, pages 159–166, 2001. [190]

F. Han and S. Zhu. Bottom-up/top-down image parsing by Attribute Graph grammar. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1778–1785, 2005. [19]

Z. S. Harris. *Methods in structural linguistics*. University of Chicago Press, Chicago, IL, 1951. [40]

M. D. Hauser, N. Chomsky, and W. T. Fitch. The faculty of language: what is it, who has it, and how did it evolve? *Science*, 298(5598):1569–1579, 2002. [23, 24]

J. Hawkins and S. Blakeslee. *On intelligence*. Henry Holt and Company, New York, 2004. [1, 11, 12, 14, 15, 16, 88, 89, 93, 103, 113, 179, 183, 185]

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. [84]

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79 (8):2554–8, 1982. [123]

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. [65]

D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology*, 195(1):215–243, 1968. [13]

J. Hummel and I. Biederman. Dynamic binding in a neural network for shape recognition. *Psychological Review*, 99(3):480–517, 1992. [25, 71]

J. Hummel and K. J. Holyoak. Distributed representations of structure: a theory of analogical access and mapping. *Psychological Review*, 104(3):427–466, 1997. [71, 90, 115]

J. Hummel and K. J. Holyoak. A symbolic-connectionist theory of relational inference and generalization. *Psychological Review*, 110(2):220–264, 2003. [115]

J. Hummel, K. J. Holyoak, K. Green, C. Doumas, and L. Devnich. A solution to the binding problem for compositional connectionism. In S. D. Levy and R. Gayler, editors, *Compositional connectionism in cognitive science: Papers from the AAAI Fall Symposium*, pages 31–34. 2004. [71, 110, 177]

E. M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, (18):245–282, 2006. [118]

E. M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452, 2007. [174]

R. Jackendoff. *Foundations of Language*. Oxford University Press, Oxford, UK, 2002. [3, 4, 20, 29, 90, 179, 183]

H. Jacobsson. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Computation*, 17(6):1223–1263, 2005. [84]

M. H. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632, 1998. [49]

A. K. Joshi. Starting with complex primitives pays off: complicate locally, simplify globally. *Cognitive Science*, 28(5):637–668, 2004. [49, 117]

D. Jurafsky and J. H. Martin. *Speech And Language Processing. An Introduction To Natural Language Processing, Computational Linguistics, And Speech Recognition. 2nd Edition.* Prentice-Hall, Englewood Cliffs, NJ, 2009. [44]

P. Kay and C. J. Fillmore. Grammatical constructions and linguistic generalizations: the What's X Doing Y? construction. *Language*, 1999. [36]

D. Klein and C. D. Manning. A generative Constituent-Context Model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia*, pages 128–135. 2002. [55, 191]

D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, 2003. [49, 130]

E. Kobatake and K. Tanaka. Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex. *Journal of Neurophysiology*, 71(3):856–67, 1994. [12]

C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4(4):219–27, 1985. [118]

T. Kohonen. *Self-organization and associative memory: 3rd edition*. Springer Verlag, Berlin, 1998. [26, 115]

T. Koskela, M. Varsta, K. Heikkonen, and K. Kaski. Time series prediction using recurrent SOM with local linear models. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 2(1):60–68, 1998. [84]

G. Kreiman, I. Fried, and C. Koch. Single-neuron correlates of subjective vision in the human medial temporal lobe. *Proceedings of the national academy of sciences*, 99(12):8378–8383, 2002. [13]

B. Kröse and P. van der Smagt. *An introduction to neural networks,8th edition*. University of Amsterdam, Amsterdam, the Netherlands, 1996. [65]

G. Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press, Chicago, IL, 1987. [59]

G. Lakoff and M. H. Johnson. *Philosophy in the Flesh*. Basic Books, New York, 1999. [36]

V. A. F. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571–579, 2000. [28]

V. A. F. Lamme and H. Spekreijse. Neuronal synchrony does not represent texture segregation. *Nature*, 78(396):362–66, 1998. [22]

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990. [56, 185]

K. Lashley. The problem of serial order in behavior. In L. Jeffress, editor, *Cerebral mechanisms in behavior*. 1951. [118]

D. A. Leopold and N. K. Logothesis. Activity changes in early visual cortex reflect monkeys' percepts during binocular rivalry. *Nature*, 1996. [86]

R. Levy. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177, 2008. [190]

W. Levy. A sequence predicting CA3 is a flexible associator that learns and uses context to solve hippocampal-like tasks. *Hippocampus*, 6:579–590, 1996. [32, 33, 123, 138]

E. Lieven, H. Behrens, J. Speares, and M. Tomasello. Early syntactic creativity: a usage-based approach. *Journal of Child Language*, 30(02):333–370, 2003. [x, 165, 168]

I. Lifshitz. *Image interpretation using bottom-up top-down cycle on fragment trees*. PhD thesis, 2005. [19]

D. MacKay. Amnesic HM exhibits parallel deficits and sparing in language and memory: Systems versus binding theory accounts. *Language and cognitive processes*, 22(3):377–452, 2007. [138]

B. MacWhinney. *The CHILDES project: Tools for analyzing talk. Third Edition*. Lawrence Erlbaum Associates, Mahway, NJ, 2000. [51, 108, 190]

C. D. Manning. Probabilistic syntax. In R. Bod, J. Hay, and S. Jannedy, editors, *Probabilistic linguistics*, pages 289–341. MIT Press, Cambridge, MA, 2003. [55, 60, 61]

C. D. Manning and B. Carpenter. Probabilistic parsing using left corner language models. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, San Francisco, CA, 1997. Morgan Kaufmann. [47, 111, 126, 130, 132, 141]

C. D. Manning and H. Schütze. *Foundations of Statistical Language Processing*. MIT Press, Cambridge, MA, 2000. [46, 60, 131]

G. F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, Cambridge, MA, 2001. [x, 4, 86, 87, 88, 93, 122, 165]

G. F. Marcus, S. Vijayan, S. Bandi Rao, and P. Vishton. Rule learning by seven-month-old infants. *Science*, 283(5398):77, 1999. [x, 87, 90, 165]

M. P. Marcus. The acquisition of the English past tense in children and multi-layered connectionist networks. *Cognition*, 56(3):271–279, 1995. [66]

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. [46, 131]

T. M. Martinetz and K. Schulten. A neural-gas network learns topologies. In T. Kohonen and K. Makisara, editors, *Artificial Neural Networks*. Elsevier, Amsterdam, 1991. [105]

J. H. R. Maunsell and W. T. Newsome. Visual processing in monkey extrastriate cortex. *Annual Review of Neuroscience*, 10(1):363–401, 1987. [13]

M. R. Mayberry III and R. Miikkulainen. SARDSRN: a neural network shift-reduce parser. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99) (San Francisco, CA)*, volume 16, pages 820–827, San Francisco, CA, 1999. Morgan Kaufmann. [26, 84]

J. L. McClelland, B. McNaughton, and R. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419, 1995. [123, 172, 176, 178]

J. L. McClelland, M. M. Botvinick, D. C. Noelle, D. Plaut, and T. T. Rogers. Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, 14(8):348–356, 2010. [64]

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943. [64]

T. McQueen. *STORM: an Unsupervised Connectionist Model for Language Acquisition*. PhD thesis, Nottingham Trent University, 2005. [25, 123]

T. A. McQueen, J. Hopgood, J. F. Allen, and J. Tepper. Extracting finite structure from infinite language. *Knowledge Based Systems*, 18:135–141, 2005. [25, 26, 84]

B. Merker. Cortex, countercurrent context, and dimensional integration of lifetime memory. *Cortex*, 40(3), 2004. [28]

R. Miikkulainen. Text and discourse understanding: The DISCERN system. In R. Dale, H. Moisl, and H. Somers, editors, *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*. Marcel Dekker, New York, 1999. [123]

R. Miikkulainen, J. A. Bednar, Y. Choe, and J. Sirosh. Self-organization, plasticity, and low-level visual phenomena in a laterally connected map model of the primary visual cortex. In R. Goldstone, P. Schyns, and D. L. Medin, editors, *Psychology of learning and motivation*, volume 36, pages 257–308. 1997. [26]

T. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers Computer Science Department, 1980. CBM-TR-117. [77, 79, 82]

T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997. [77]

G. Mongillo, O. Barak, and M. Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):1543, 2008. [25, 116]

R. C. Moore. Improved left-corner chart parsing for large context-free grammars. pages 185–201, 2004. [126]

R. G. Morris and U. Frey. Hippocampal synaptic plasticity: role in spatial learning or the automatic recording of attended experience? *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1360): 1489–1503, 1997. [32]

V. B. Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4): 701–722, 1997. [13, 26, 97]

L. Nadel and M. Moscovitch. Memory consolidation, retrograde amnesia and the hippocampal complex. *Current opinion in neurobiology*, 7(2):217–227, 1997. [176]

L. Nadel, A. Samsonovich, L. Ryan, and M. Moscovitch. Multiple Trace Theory of human memory: computational, neuroimaging, and neuropsychological results. *Hippocampus*, 10(4):352–368, 2000. [177]

J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1): 171–5, 1971. [33]

J. O'Keefe and L. Nadel. The hippocampus as a cognitive map. *Behavioral and Brain Sciences*, 2(4):487–494, 1979. [33, 175]

B. Opitz. Neural binding mechanisms in learning and memory. *Neuroscience & Biobehavioral Reviews*, 34(7):1036–1046, 2010. [34]

R. O'Reilly and K. Norman. Hippocampal and neocortical contributions to memory: advances in the complementary learning systems framework. *Trends in Cognitive Sciences*, 6(12):505–510, 2002. [123, 138, 176]

R. O'Reilly and J. W. Rudy. Conjunctive representations in learning and memory: Principles of cortical and hippocampal function. *Psychological Review*, 108(2): 311–45, 2001. [138, 172, 176, 177]

B. Palanca and G. C. DeAngelis. Does neuronal synchrony underlie visual feature grouping? *Neuron*, 46(2):333–346, 2005. [22]

B. H. Partee, A. ter Meulen, and R. E. Wall. *Mathematical methods in linguistics*. Kluwer Academic Publishers, Boston, MA, 1990. [43]

C. S. Peirce. Logic as semiotic: The theory of signs. In R. E. Innis, editor, *Semiotics: An introductory anthology*, pages 4–23. Bloomington, IN: Indiana University Press, Indiana, 1903. [68]

A. Peters. *The units of language acquisition*. Cambridge University Press, Cambridge, UK, 1983. [5]

S. Petrov and D. Klein. Learning and inference for hierarchically split PCFGs. In *Proceedings of the 22nd national conference on Artificial intelligence*, volume 22, page 1663, 2007. [49]

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 443–440, 2006. [23, 61, 129]

S. Pinker and A. Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28:73–193, 1988. [66]

K. Plunkett and V. A. Marchman. Learning from a connectionist model of the acquisition of the English past tense. *Cognition*, 61(3):299–308, 1996. [66]

D. Poeppel, W. J. Idsardi, and V. van Wassenhove. Speech perception at the interface of neurobiology and linguistics. *Philosophical Transactions of the Royal Society London*, 363(1493):1071–86, 2008. [28]

J. B. Pollack. Recursive auto-associative memory. *Neural Networks*, 1:122, 1988. [8, 67, 69, 75, 93, 110]

J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 46 (1-2):77–105, 1990. [69]

M. Pollack. *On connectionist models of natural language processing*. PhD thesis, University of Illinois, Urbana, 1987. [72]

D. Prescher. A tutorial on the Expectation-Maximization algorithm including maximum-likelihood estimation and EM training of probabilistic context-free grammars. 2003. [46]

A. Prince and P. Smolensky. Optimality: from neural networks to universal grammar. *Science*, 275(5306):1604–10, 1997. [5, 8, 69, 70, 93]

F. Pulvermüller. Words in the brain's language. *Behavioral and Brain Sciences*, 22(2):253–279, 1999. [71]

R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435:1102–7, 2005. [13]

R. L. Redondo and R. G. Morris. Making memories last: the synaptic tagging and capture hypothesis. *Nature Reviews Neuroscience*, 12(1):17–30, 2010. [174]

R. Reichart and A. Rappoport. Unsupervised induction of labeled parse trees by clustering with syntactic features. In *Proceedings of the 8th International Conference on Computational Linguistics (COLING 1992) Nantes, France.*, pages 721–728, 1992. [191]

H. Ritter and T. Kohonen. Self-organizing semantic maps. *Biological Cybernetics*, 61(4):241–254, 1989. [26, 27]

P. Rodriguez, J. Wiles, and J. L. Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999. [84, 200, 201]

P. R. Roelfsema. Cortical algorithms for perceptual grouping. *Annual Review of Neuroscience*, 29:203–227, 2006. [13, 22, 24, 25, 28, 90]

P. R. Roelfsema and H. Spekreijse. Binding contour segments into spatially extended objects. In *Neurobiology of Attention*, Amsterdam, 2005. Elsevier. [22, 23]

P. R. Roelfsema, V. A. F. Lamme, and H. Spekreijse. Synchrony and covariation of firing rates in the primary visual cortex during contour grouping. *Nature Neuroscience*, 7(9):982–991, 2004. [22]

E. Rosch. Principles of categorization. In E. Rosch and B. B. Lloyd, editors, *Principles of categorisation, in cognition and categorisation*, pages 27–48. Erlbaum, Hillsdale, NJ, 1978. [59]

E. Rosch and C. B. Mervis. Family resemblances: studies in the internal structure of categories. *Cognitive Psychology*, 7(4):573–605, 1975. [59]

D. Rosenkrantz and P. Lewis II. Deterministic left corner parsing. In *11th Annual Symposium on Switching and Automata Theory*, pages 139–152, New York, 1970. IEEE Press. [18, 103, 126]

J. R. Ross. *Constraints on Variables in Syntax*. PhD thesis, MIT, 1967. [59]

D. E. Rumelhart and J. L. McClelland. On learning the past tenses of English verbs. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol. 2*, pages 318–362. MIT Press, Cambridge, MA, 1986. [66]

J. R. Saffran, R. Aslin, and E. L. Newport. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–8, 1996. [83]

K. Sagae, E. Davis, A. Lavie, B. MacWhinney, and S. Wintner. High-accuracy annotation and parsing of CHILDES transcripts. In *Proceedings of the ACL-2007 Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 25–32, 2007. [51]

F. Sangati and W. Zuidema. Accurate parsing with compact tree-substitution grammars: Double-DOP. In *Proceedings of the 2011 Joint Conference on Empirical Methods in Natural Language Processing*, 2011. [188]

F. Sangati, W. Zuidema, and R. Bod. A generative re-ranking model for dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 238–241, -, 2009. Association for Computational Linguistics. [131]

R. Scha. Taaltheorie en taaltechnologie: competence en performance. In *Computertoepassingen in de Neerlandistiek, Almere: Landelijke Vereniging van Neerlandici (LVVN-jaarboek)*, volume 11 of *7–22*, pages 409–440. 1990. [30]

R. Scha, R. Bod, and K. Sima'an. A memory-based model of syntactic analysis: data-oriented parsing. *Journal of experimenal and theoretical artificial intelligence*, 11(3):409–440, 1999. [5, 30, 49]

J. Schmidhuber. The neural heat exchanger. In *Progress in Neural Information Processing: Proceedings of the Intl. Conference on Neural Information Processing*, -, 1996. Springer Verlag. [28]

W. B. Scoville and B. Milner. Loss of recent memory after bilateral hippocampal lesions. *Journal of Neurology, Neurosurgery & Psychiatry*, 20(1):11, 1957. [32]

D. Servan-Schreiber, A. Cleeremans, and J. L. McClelland. Graded state machines: the representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7(2):161–193, 1991. [84]

L. Shastri. Episodic memory and cortico-hippocampal interactions. *Trends in Cognitive Sciences*, 6(4):162–168, 2002. [32, 125, 138, 177]

S. M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343, 1985. [44]

H. Siegelmann and E. Sontag. Turing computability with neural nets. *Applied Mathematics letters*, 4(6):77–80, 1991. [72]

W. Singer and C. M. Gray. Visual feature integration and the temporal correlation hypothesis. *Annual Review of Neuroscience*, 18(1):555–586, 1995. [22, 115]

M. Smets. A U-DOP approach to modeling language acquisition, 2010. MSc Thesis. [168]

P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990. [70, 75]

P. Smolensky and G. Legendre. *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*. MIT Press, Cambridge, MA, 2006. Volume 1: Cognitive Architecture. [67, 70, 110]

L. R. Squire and X. Alvarez. Retrograde amnesia and memory consolidation: a neurobiological perspective. *Current opinion in neurobiology*, 5:169–177, 1995. [32, 176]

L. Steels. Perceptually grounded meaning creation. In M. Tokoro, editor, *Proceedings of the International Conference on Multiagent Systems (ICMAS-96)*, pages 338–344, Menlo Park, CA, 1996. AAAI Press/MIT Press. [37, 83]

L. Steels. The origins of syntax in visually grounded robotic agents. *Artificial Intelligence*, 103(1):133–156, 1997. [37]

L. Steels. *The Talking Heads Experiment. Words and Meanings*. VUB, Brussels, Belgium, 1999. [37]

L. Steels. Constructivist development of grounded Construction Grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona*. Morgan Kaufmann, San Francisco, CA, 2004. [37]

K. E. Stephan, C. C. Hilgetag, G. A. P. C. Burns, M. A. O'Neill, and M. P. Young. Computational analysis of functional connectivity between areas of primate cerebral cortex. *Philisophical Transactions of the Royal Society, London, Series B.*, 355(1393):111–126, 2000. [92]

T. Stewart and C. Eliasmith. Compositionality and biologically plausible models. In M. Werning, W. Hinzen, and E. Machery, editors, *Oxford Handbook of Compositionality*. Oxford University Press, Oxford, UK, 2009. [71]

A. Stolcke. Syntactic category formation with vector space grammars. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, pages 908–912, Mahway, NJ, 1991. Lawrence Erlbaum Associates. [99]

A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California, Berkeley, Dept. of Electrical Engineering and Computer Sciences, 1994. Dept. of Electrical Engineering and Computer Sciences. [57, 80, 166, 187]

A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. In *Proceedings of the 33d Annual Meeting of Association for Computational Linguistics (ACL 1995)*, 1995. [53, 54, 55, 140, 142, 145, 147, 148]

A. Stolcke and S. M. Omohundro. Inducing probabilistic grammars by Bayesian Model Merging. In *Proceedings of the Second International Colloquium on Grammatical Inference and Applications (ICGI'94)*, volume 862 of *Lecture Notes in Computer Science*, pages 106–118, Berlin, 1994. Springer Verlag. [16, 57, 105, 166, 187, 191]

P. Sturt, F. Keller, and A. Dubey. Syntactic priming in comprehension: Parallelism effects with and without coordination. *Journal of Memory and Language*, 62(4):333–351, 2010. [190]

G. R. Sutherland and B. McNaughton. Memory trace reactivation in hippocampal and neocortical neuronal ensembles. *Current opinion in neurobiology*, 10(2): 180–186, 2000. [32]

W. Tabor. Fractal encoding of context-free grammars in connectionist networks. *Expert Systems*, 17(1):41–56, 2000. [84]

M. Tomasello. Do young children have adult syntactic competence? *Cognition*, 74:209–253, 2000a. [5, 31, 60]

M. Tomasello. The item-based nature of children's early syntactic development. *Trends in Cognitive Sciences*, 4(4):156–163, 2000b. [60, 122]

M. Tomasello. First steps toward a usage-based theory of language acquisition. *Cognitive Linguistics*, 11(1-2):61–82, 2001. [5, 31, 60, 106]

M. Tomasello. *Constructing a Language: A Usage-Based Theory of Language Acquisition.* Harvard University Press, Cambridge, MA, 2003. [5]

M. Tomasello. *Constructing a Language : A Usage-Based Theory of Language Acquisition.* Harvard University Press, -, 2005. [52, 61, 181]

D. Tse, R. F. Langston, M. Kakeyama, I. Bethus, and P. A. Spooner. Schemas and memory consolidation. *Science*, 316(5821):76, 2007. [176]

E. Tulving. Episodic and semantic memory. In E. Tulving and W. Donaldson, editors, *Organization of memory*, pages 381–402. Academic Press, New York, 1972. [30]

M. Ullman. Sequence-seeking and counter streams: A model for information processing in the cortex. Technical report, MIT, 1991. [18, 28]

M. Ullman, S. Corkin, M. Coppola, Hickok, and J. Growdon. A neural dissociation within language: Evidence that the mental dictionary is part of declarative memory, and that grammatical rules are processed by the procedural system. *Journal of Cognitive Neuroscience*, 9(2):266–276, 1997. [31]

S. Ullman. Object recognition and segmentation by a fragment-based hierarchy. *Trends in Cognitive Sciences*, 11(2):58–64, 2007. [19, 20, 103]

S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *nature neuroscience*, 5(7):682–687, 2002. [19]

F. van der Velde and M. de Kamps. Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, 29(1):37–69, 2006. [71, 72, 86, 90, 184]

F. van der Velde, G. T. van der Voort van der Kleij, and M. de Kamps. Lack of combinatorial productivity in language processing with simple recurrent networks. *Connection Science*, 16(1):21–46, 2004. [22, 75]

T. van Gelder. The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, 1998. [27]

J. van Kampen. The learnability of syntactic categories. In J. van Kampen and S. Baauw, editors, *Proceedings of GALA 2003*, pages 245–256. 2003. [5, 60]

D. van Uytsel, F. van Aelten, and D. van Compernolle. A structured language model based on context-sensitive probabilistic left-corner parsing. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8, 2001. [139, 140, 146, 149, 158, 188]

T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15(8-9):979–991, 2002. [26, 84]

C. von der Malsburg. The correlation theory of brain function. In E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of neural networks II*, pages 1–38. 1981. Internal report. [22]

T. Vosse and G. Kempen. Syntactic structure assembly in human parsing: a computational model based on competitive inhibition and a lexicalist grammar. *Cognition*, 75(2):105–143, 2000. [117]

K. Wexler. Innateness of language. In R. A. Wilson and F. C. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*, pages 408–409. MIT Press, Cambridge, MA, 1999. [59]

J. G. Wolff. Language acquisition, data compression and generalization. *Language and Communication*, 2(1):57–89, 1982. [56]

C. C. Woodruff, J. D. Johnson, and M. R. Uncapher. Content-specificity of the neural correlates of recollection. *Neuropsychologia*, 43(7):1022–1032, 2005. [32]

M. P. Young. The organization of neural systems in the primate cerebral cortex. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 252 (1333):13–18, 1993. [92]

S. Zeki. *A Vision of the Brain*. Blackwell, -, 1993. [21]

W. Zuidema. What are the productive units of natural language grammar? a DOP approach to the automatic identification of constructions. In *Proceedings of the 10th International Conference on Computational Natural Language Learning (CONLL-X)*, pages 29–36, 2006. [50, 51]

W. Zuidema. Parsimonious data-oriented parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007. [50, 51, 190]

A. Zylberberg, D. F. Slezak, P. R. Roelfsema, S. Dehaene, and M. Sigman. The brains router: A cortical network model of serial processing in the primate brain. *PLOS computational biology*, 6(4), 2010. [118]

Index

abstraction
  in HPN, 106
  in language acquisition, 5, 31
  neural basis of, 89
activation, *see* trace activation
activation function, 64
addressor system, *see* switchboard construction
All-Fragments Grammar, 159
analogical inference
  in language learning, 164–169
artificial language learning, 165
attach operation, 44, 144
attention
  in serial binding, 23
automaton, 42

back-off smoothing, 130
Bayesian Model Merging, 16, 56
  and HPN, 187
beam search, 148
binarization, *see* Markovization
binding
  by synchrony, 22
  conjunctive, 21, 71
  contour binding, 21
  dynamic
    in HPN, 98, 113–116
    in syntax, 90–91
    in vision, 22
    of episodic memories, 173
  fillers to roles, 69, 70
  serial binding, 22, 184
  with tensor product, 70
binding neuron, 138
binding probability, *see* HPN, binding probability
binding problem
  in language, 20, 90
  in vision, 21
  massiveness of, 4, 71, 184
bottom-up network, *see* countercurrent systems

catastrophic interference, 176
categorization
  in HPN, 106–181
  in language, 59
category
  gradedness of, 59–61
  prototypical, 59
  syntactic, 40
chart parsing, 52
  left corner, *see* parsing, Left Corner Chart Parsing
Chomsky hierarchy, 42–44
  and systematicity, 78–83
  place of natural language in, 43
chunk operator, 17, 57
Clinton cell, *see* grandmother cells

cognitive map, *see* relational network
column, *see* cortical column
common history, 128
  updating, 150
complementary learning systems framework, 176
complete state, 53, 140
complex cells, 13
complex unit, *see* treelet
compressor node, *see* HPN, compressor node
conceptual pole, *see also* cortical column
connectionism, 4, 64
connectionist constraint, 68
  violation of, 70, 71
connectionist networks
  distributed, 86, 89
  localist, 26
  recurrent, 72
constituent, 40
constituent structure, 66
  encoding in SRN, 198
constructicon, 5, 61
  implementation in HPN, 181
construction, 5, 29
construction grammar, 5, 36, 61
  and HPN, 182
content addressability, 123
  in Episodic Grammar, 137
context
  structural and lexical, 48, 111
context dependence
  in syntax, 48
  of episodic memory, 30
context free grammar, *see* grammar, context free
context invariant
  criterion, *see* systematicity, context invariance criterion for
  operations, 25
cortex
  computation in the, 12
  prediction in the, 12
cortical column, 13
countercurrent systems, 28, 90

Data Oriented Parsing, 49, 135
  relation to Episodic Grammar, 136
declarative memory, 30
derivation
  in Episodic Grammar, 125
  in HPN, 101
  left corner, 48
  left-most, 41
dynamic binding, *see* binding, dynamic
dynamic programming, 52

Earley parser, 52–54, 140
elementary tree, 49
EM algorithm, *see* Expectation Maximization Algorithm
encapsulation
  in HPN, 109
  in neural networks, 88
  in the brain, 16
Episodic Grammar, 123–126
episodic left corner chart parser, *see* parsing, episodic Left Corner Parsing
episodic left corner grammar, *see* Left Corner Shifting Grammar, episodic
episodic memory, 30
  decontextualization of, 33, 172
  properties of, 122
  retrieval
    in Episodic Grammar, 125
episodic shortest derivation parser, 154
episodic-HPN, 169–171
estimation
  maximum likelihood, 46
  relative frequency, 46
Expectation Maximization Algorithm, 55

F-score, 47

feedforward network, *see* multi-layer perceptron
finite state automaton, 42
formal grammar, 40
forward probability, 54
  in left corner parsing, 142

generalization
  in SRN, 200
  inductive bias for, 77–78
generative power
  of formal grammars, 42
Gestalt laws of perception, 85
goal state, 140
Goodman Reduction, 159
graded categories, 59–60
  and topology, 27
  in HPN, 97
grammar
  context free, 41
  context sensitive, 42
  generative, 58
  phrase structure, 41
  probabilistic context free, 45
  regular, 42
  rewriting, 40
  Tree Substitution, *see* Tree Substitution Grammar
  Usage Based, *see* Usage Based Grammar
grammar induction, unsupervised, 55–58
grandmother cells, 13
grounded semantics, 37

hidden unit, 65
Hierarchical Prediction Network (HPN), 95–119
hippocampus, 32
  as a switchboard, 174
  as fast learning system, 176
  replay by the, 32
  role in episodic-HPN, 172
HPN
  binding probability, 99
  compressor node, 97
    neural correlate, 117
  conversion from CFG, 99, 205–206
  input node, 97
  node state, 103, 116
  path connector, 101, 104
  slot, 97
  stack implementation, 101
  substitution space, 97, 113
    as relational network, 174

implicit knowledge, 66
  of constituent structure, 199
  of word categories, 196
independence assumptions, 45, 88
inductive bias, *see* generalization, inductive bias for
inner probability, 54
  in left corner parsing, 142
input node, *see* HPN, input node
Inside Outside algorithm, 56, 185
invariants
  in HPN, 109
  versus variables, 89, 183

Kohonen network, 26

labeled precision, 47
labeled recall, 47
language model, 48
  in episodic-HPN, 171
learning
  associative versus discriminative, 83
  from analogy, 78, *see also* analogical inference, in language learning, 165
  in episodic-HPN, 171
  in HPN, 104–105
  locality of, 86
Left Corner Shifting Grammar, 141

episodic, 126
probabilistic, 141
left span index, 141
lexicalization, 48

Markovization
horizontal and vertical, 49
in Episodic Grammar, 130
McCulloch-Pitts neuron, 64
meaning
local versus global scope, 68
memory
episodic, *see* episodic memory
semantic, *see* semantic memory
short-term, 25, 116
memory consolidation, 32, 34, 164
connectionist models of, 175–177
in episodic-HPN, 172
in language acquisition, 164–167
Memory Prediction Framework, 11–16, 89, 113
memory space, *see* relational network
merge operator, 17, 57
Minimum Description Length
principle of, 58, 167
multi-layer perceptron, 65
multi-word constructions, *see* construction
multiple trace model, 176

name cells, 16, 89, 113
navigational map, 33
neural assembly
syntactic, 17
neural blackboard, 71
neural networks, *see* connectionist networks
nonterminal, 40

object recognition
fragment-based, 19
in the brain, 18
orientation column, 13

paradigmatic process, 17
parent annotation, 49
parse tree, 40, 44
in HPN, 101
parser, *see* parsing, 44
PARSEVAL metric, 46
parsing, 18
bottom-up, 44
episodic Left Corner Parsing, 149
history based, 49, 136
in episodic-HPN, 170
Left Corner Chart Parsing, 142
Left Corner Parsing, 18, 44
in HPN, 103, 203
probabilistic, 47
top-down, 44
visual parsing, 19
with shortest derivation, 154
path connector, *see* HPN, path connector
PCFG, *see* grammar, probabilistic context free
Penn Treebank corpus, 46
perceptual pole, *see also* cortical column
phrase structure
encoding in SRN, 198
grammars, 2
tree, *see* parse tree
place cells, 33
pointer
implementation in HPN, 98
implementation in the brain, 89
Pollack, P., 69
prediction, *see* cortex, prediction in the
prefix probability, 55
in left corner parsing, 145
priming, 33
in Episodic Grammar, 125
prioritized queue, 148
problem of two, 4, 184

productivity of language, 65, *see also* systematicity
project operation, 44, 143

RAAM, 69
  and HPN, 110
Radical Construction Grammar, 5, *see also* construction grammar, 60, 61
recurrent neural networks, *see* connectionist networks, recurrent
recursion, 23
  leaky, 79
Recursive Auto-Associative Memory, *see* RAAM
recursive systematicity, 77
  of context free grammars, 81
  of HPN, 110
  of SRN, 84
register, 93, 161
register state, 127
reinstatement hypothesis of episodic memory retrieval, 32, 126
relational network, 33
  and substitution space, 174
representational power, *see also* generative power
  of HPN, 99, 205
  of recurrent neural networks, 72
reranking, 131
rewrite rule, 2, 40
right span index, 141

scanned state, 145
Self-Organizing Map, *see* Kohonen network
semantic memory, 30
  and HPN, 122
semantics, 36–38
shift
  operation, 44, 145
  probability, 142
  rule, 141
  space, 171
  treelet, 127
short-term memory, *see* memory, local short-term
shortest derivation
  as simplicity bias, 168
  parse, 152
  reranking with, 135
Shortest Derivation Length (SDL), 152
Simple Recurrent Network, 73, 83, 195–202
simplicity bias, 167
Simplicity-DOP, 161
slot
  in construction, 29, 31
  in HPN, 97
smoothing
  in Episodic Grammar, 129
stack
  and working memory, 116
  in HPN, *see* HPN, stack implementation
standard consolidation model, 176
starred nonterminal, 127
state
  in HPN, *see* HPN, node state
  in left corner parsing, 140
  of automaton, 42
  of Earley parser, 52
string probability, 54
strong systematicity, 75
  of context free grammars, 80
  of SRN, 84, 196
structure
  ambiguity of, 45
  dependence of, 66
  in language, 37
structure bias, 78
  of formal grammars, 79
  of the SRN, 83
substitutability (HPN), 98
substitution, 24
  neural correlate of, 24

substitution space, *see* HPN, substitution space
surprisal theory, 190
switchboard construction, 91
  neural implementation, 113
  role of hippocampus, 174
symbol, 68
synaptic tagging and capture hypothesis, 174
syntagmatic process, 17
syntax, 66
  and systematicity, 83
  neural theory of, 35–36
systematicity
  and the Chomsky hierarchy, 78–83
  concise criteria, 76
  context invariance criterion for, 76
    applied to HPN, 109
    applied to SRN, 85
  Hadley's criteria for evaluating, 75
  of connectionist networks, 74
  of language, 66

tag, 24, *see also* tagging system
tagging system, 115, *see also* binding, dynamic, in syntax
tensor product, *see* binding, tensor product
terminal, 40
tie-breaking, 170
top-down network, *see* countercurrent systems
top-down prediction, *see* cortex, prediction in the
topographic map, *see* topology
topological self-organization, 26
  experiment in HPN, 107
  in switchboard, 92
topology, 13
  in language acquisition, 61
  virtual syntactic, 91
trace
  in Episodic Grammar, 123
  of episodic memory, 122
trace activation, 128
transition cost, 152
Tree Substitution Grammar, 49
treebank, 46
treelet, 93
  state, 150
  type, 150

U-DOP, 168, 188
Universal Grammar, 59
Usage Based Grammar, 5, 60
  and HPN, 181

variable
  binding, *see* binding, fillers to roles
  operations, 3, 66
  representation in neural networks, 3, 183
ventral pathway, 12
verb islands, 60, *see also* Usage Based Grammar
visual hierarchy, 12
Viterbi parse, 55
  in Episodic Grammar, 143
Viterbi predecessor state, 55
Viterbi probability, 55, 143

weak systematicity, 75
winner-take-all effect, 26
word prediction task, 74, 195
working memory, 25, 116

Samenvatting

In dit proefschrift onderzoek ik de neurale mechanismen, die ten grondslag liggen aan het menselijk vermogen tot het leren, onthouden en gebruiken van grammaticale structuur in taal, de syntax. Daarbij gebruik ik inzichten uit de taalkunde, de cognitieve psychologie en de neurobiologie.

Uit taalkundig onderzoek blijkt dat men de structuur van vrijwel alle talen, kort samengevat, als volgt kan karakteriseren: taal is produktief – met een beperkt aantal woorden en taalregels kunnen wij een onbeperkt aantal nieuwe zinnen maken en begrijpen. Daarnaast is taal hiërarchisch – zinnen zijn opgebouwd uit zinsdelen, welke laatste weer kunnen zijn opgebouwd uit andere zinsdelen, etc. Deze twee kenmerken leveren meteen de minimum voorwaarden waaraan een systeem voor taalverwerking, zoals het menselijk brein, moet voldoen. Een eerste bijdrage van dit proefschrift is om deze voorwaarden zo precies mogelijk te formuleren, zodat verschillende theorieën over informatieverwerking in het brein (zogenaamde neurale netwerken) hieraan kunnen worden getoetst. Hieruit blijkt dat bestaande klassen van neurale netwerken (in het bijzonder de zogenaamde gedistribueerde recurrente netwerken) ongeschikt zijn voor het modelleren van taal, vanwege bepaalde oversimplificerende aannames.

In de rest van het proefschrift ontwikkel ik daarom een nieuwe neurale theorie van syntax, die wel rekening houdt met de hiërarchische structuur en productiviteit van taal. Daarbij heb ik mij laten inspireren door Jeff Hawkins' "Memory Prediction Framework" (MPF). Dit is een theorie over het brein die stelt dat de belangrijkste functie van de neocortex (een deel van de hersenschors) het voorspellen van nieuwe situaties en het anticiperen op toekomstige acties is. Volgens Hawkins slaat de neocortex daartoe alle informatie en kennis op als temporele sequenties, met een hiërarchische structuur. Knopen, die hoger liggen in de hiërarchie, representeren abstractere begrippen en langere tijdspannes (temporele compressie).

Terwijl Hawkins zijn theorie vooral heeft uitgewerkt voor visuele waarneming, benadruk ik de analogieën met taal: ook syntactische categorieën vertegenwoor-

digen temporele sequenties (namelijk van woorden); wanneer deze in een vroeg stadium van een zin herkend worden helpen ze het verdere verloop van de zin te voorspellen. Ik stel voor dat syntactische categorieën, net als visuele categorieën, lokaal in het brein zijn geëncodeerd in corticale kolommen, en bovendien dat de hiërarchische en topologische organisatie van al deze 'syntactische' kolommen samen een grammatica vormt.

Een tweede inspiratiebron voor mijn onderzoek is de rol van het geheugen in taal. Een belangrijke vraag in de taalkunde is hoe zinsfragmenten (bestaande uit meerdere woorden) zijn opgeslagen in het geheugen, zodat ze als geheel kunnen worden hergebruikt in nieuwe zinnen. Voorbeelden hiervan zijn *'Hoe gaat het ermee?'*, of *'in aanmerking genomen, dat ...'*. Volgens sommige taalkundigen is in feite elke zin samengesteld uit kleinere of grotere brokstukken van andere zinnen.

Om bovenstaande vraag te kunnen beantwoorden stel ik voor dat we in de taalkunde, net als in de cognitieve psychologie, onderscheid moeten maken tussen twee soorten geheugens: een geheugen voor abstracte, relationele kennis, het zogeheten 'semantische' geheugen, en een geheugen voor individuele gebeurtenissen, geplaatst in een persoonlijke context, het zogeheten 'episodische geheugen' (bijvoorbeeld de herinnering van een verjaardagsfeestje). In dit proefschrift verdedig ik de stelling dat, terwijl abstracte taalregels en syntactische categorieën onderdeel uitmaken van het semantische geheugen, het episodische geheugen verantwoordelijk is voor het onthouden van zinsfragmenten (zoals bovengenoemde), en zelfs complete zinnen. Het episodische geheugen speelt eveneens een belangrijke rol bij het leren van taal, aangenomen dat onze talenkennis niet is aangeboren, maar voortkomt uit individuele talige ervaringen. Hoe de dagelijkse episodische ervaringen in het brein worden omgezet in abstracte, semantische kennis is een belangrijke onderzoeksvraag in de cognitieve en de neuro-wetenschappen, die niet alleen specifiek is voor taal.

In mijn onderzoek probeer ik inzicht te verkrijgen in de mechanismen van het leren en produceren van taal door te kijken naar de parallellen tussen taalprocessen en geheugenprocessen in het brein. Hiertoe formuleer ik een expliciete theorie over de wisselwerking tussen een episodisch en semantisch geheugen voor taal, het "Hierarchical Prediction Network" (HPN) genaamd, waarin tevens de ideeën van Hawkins zijn opgenomen (met enkele belangrijke wijzigingen).

Het semantische geheugen voor taal wordt in HPN voorgesteld als een neuraal netwerk, waarvan de knopen (overeenkomend met syntactische en lexicale corticale kolommen) zijn geordend in een topologische ruimte. Dit wil zeggen dat knopen die een vergelijkbare functie vervullen in de syntactische analyse van zinnen bij elkaar in de buurt liggen. (Dit is geïnspireerd op de visuele cortex, waar bijvoorbeeld kolommen voor het herkennen van de oriëntatie van een lijn segment topologisch zijn georganiseerd.) Een syntactische analyse van een zin bestaat in HPN uit een pad langs een aantal knopen in het netwerk, die 'dynamisch' (flexibel) met elkaar zijn gebonden via het centraal uitwisselen van gegevens. (Ook dit

is geïnspireerd op hoe het brein primitieve visuele categorieën bindt tot complexe vormen.) Doordat de bindingen tussen knopen flexibel zijn (in tegenstelling tot de vaste links tussen knopen in conventionele neurale netwerken) kan HPN de productiviteit van taal verklaren.

Het episodische geheugen voor taal is in HPN ingebed in het semantische geheugen, in de vorm van permanente geheugensporen, die worden achtergelaten in de knopen van het netwerk als gevolg van het verwerken van een zin. Op die manier kan de netwerk-analyse van een verwerkte zin via de geheugensporen altijd later gereconstrueerd worden. Bovendien kunnen nieuwe zinnen worden gevormd door sporen van (fragmenten van) oude zinnen te combineren.

Dit proefschrift is als volgt georganiseerd: in hoofdstuk 1 bespreek ik de doeleinden van mijn onderzoek en de motivatie voor de gekozen aanpak. In hoofdstuk 2 introduceer ik het Memory Prediction Framework, en geef ik de neurobiologische achtergrond voor de neurale theorie van syntax. In hoofdstuk 3 behandel ik enkele basisbegrippen uit de formele taalkunde en de computationele linguistiek, met speciale aandacht voor parseertechnieken, die worden gebruikt in het HPN model. Hoofdstuk 4 bevat een (kritisch) overzicht van de literatuur over neurale netwerken van taal, in de context van de discussie over de basiskenmerken van taal: productiviteit en hiërarchie.

In hoofdstuk 5 tot en met 8 ontwikkel ik, in een aantal stappen, het HPN model. Om de voorspellingen van de neurale theorie van syntax kwantitatief te kunnen toetsen, heb ik het HPN model geïmplementeerd op de computer, zodat ik simulaties kon draaien op basis van vele duizenden zinnen. In hoofdstuk 5 beschrijf ik allereerst een computer-implementatie van het basismodel zonder episodisch geheugen, en hiermee laat ik zien dat een syntactische topologie geleerd kan worden uit simpele, kunstmatig geproduceerde zinnen. Dan, in hoofdstuk 6 en 7, bespreek ik een computermodel van het episodisch geheugen voor taal, waaruit ik voor het gemak de topologie weglaat, en ik test dit op een groot aantal originele zinnen aan de hand van een taak voor zinsontleding. In hoofdstuk 8 voeg ik tenslotte alle componenten van HPN samen in een enkele implementatie, die precies beschrijft hoe uit episodische taal-ervaringen een abstracte grammatica in de vorm van een netwerk topologie wordt geconstrueerd. In dit hoofdstuk benadruk ik de gelijkenis tussen het leren van taal en het proces van geheugenconsolidatie – het omzetten in het brein van informatie bestaande uit concrete episodes naar een netwerk van abstracte, semantische kennis. In hoofdstuk 9 volgen een algemene discussie en een aantal ideeën voor toekomstig onderzoek.

De belangrijkste conclusie van mijn proefschrift is dat het mogelijk en zinvol is om inzichten uit de (computationele) taalkunde te koppelen aan neuro-biologische en cognitieve inzichten, en vice versa. Enerzijds kunnen uit de strenge functionele eisen die de taal aan de hersenen stelt een aantal niet triviale conclusies worden afgeleid over de verwerking en opslag van informatie in het brein, en anderzijds leveren de fysiologische beperkingen van de hersenen onverwachte uitdagingen voor theorieën van syntax, zoals het gebruik van topologie.

Abstract

In this dissertation I investigate the neural mechanisms underlying the human ability to learn, store and make use of grammatical structure, so-called syntax, in language. In doing so I incorporate insights from linguistics, cognitive psychology and neuro-biology.

From linguistic research it is known that the structure of nearly all languages exhibits two essential characteristics: language is productive – from a limited number of words and rules one can produce and understand an unlimited number of novel sentences. Further, language is hierarchical – sentences are constructed from phrases, which in turn can be constructed from other phrases, etc. These two structural properties of language provide minimum requirements that a system of language processing, such as the brain, must satisfy. A first contribution of this dissertation is that it attempts to formulate these requirements as concisely as possible, allowing for a strict evaluation of existing models of neural processing in the brain (so-called neural networks). From this evaluation it is concluded that conventional types of neural networks (in particular so-called recurrent, fully distributed networks) are unsuited for modeling language, due to certain oversimplifying assumptions.

In the remainder of this thesis I therefore develop a novel type of neural network, based on a neural theory of syntax that does take into account the hierarchical structure and productivity of language. It is inspired by Jeff Hawkins's Memory Prediction Framework (MPF), which is a theory of information processing in the brain that states, among other things, that the main function of the neocortex is to predict, in order to anticipate novel situations. According to Hawkins, to this end the neocortex stores all processed information as temporal sequences of patterns, in a hierarchical fashion. Cellular columns that are positioned higher in the cortical hierarchy represent more abstract concepts, and span longer times by virtue of temporal compression.

Whereas Hawkins applies his theory primarily to the area of visual perception, in my dissertation I emphasize the analogies between visual processing and lan-

guage processing: temporal compression is a typical feature of syntactic categories (as they encode sequences of words); whenever these categories are recognized in an early stage of the sentence, they can be expanded to predict the subsequent course of the sentence. I propose therefore that syntactic categories, like visual categories, are represented locally in the brain within cortical columns, and moreover that the hierarchical and topological organization of such 'syntactic' columns constitutes a grammar.

A second source of inspiration for my research is the role of memory in language processing and acquisition. An important question that a neural theory of language has to address concerns the nature of the smallest productive units of language that are stored in memory. When producing a novel sentence it seems that language users often reuse entire memorized sentence fragments, whose meanings are not predictable from the constituent words. Examples of such multiword constructions are '*How do you do?*' or '*kick the bucket*', but there are also productive constructions with one or more open 'slots', such as '*the more you think about X, the less you understand*', or completely abstract and unlexicalized constructions. According to certain linguistic theories every sentence in a language can be formed by combining constructions of varying degrees of complexity and abstractness.

In order to answer the question about the storage of constructions I propose that in linguistics, as in cognitive psychology, one must distinguish between two kinds of memory systems: a memory system for abstract, relational knowledge, so-called 'semantic' memory, and a memory system for personally experienced events or 'episodes' (for instance the memory of a birthday party), embedded in a temporal and spatial context, so-called 'episodic' memory. I contend that, while abstract rules and syntactic categories of a language are part of a semantic memory for language, an episodic memory is responsible for storing sentence fragments, and even entire sentences.

Episodic memory also plays an important role in language acquisition, assuming that our linguistic knowledge is not innate, but originates from the assimilation of many individual linguistic experiences. An important claim of this thesis is that language acquisition, like knowledge acquisition in other cognitive domains, can be understood as a gradual transformational process of concrete episodic experiences into a system of abstract, semantic memories.

Starting from the assumption that universal mechanisms of memory processing in the brain also govern language production and acquisition, I formulate an explicit theory about the interaction between an episodic and a semantic memory for language, called the "Hierarchical Prediction Network" (HPN), that is applied to sentence processing and acquisition. HPN further incorporates the ideas of the MPF, with some important modifications.

The semantic memory for language is conceived of in HPN as a neural network, in which the nodes (corresponding to syntactic and lexical cortical columns) derive their function from their topological arrangement in the network. This means

that two nodes that fulfill a similar function within the syntactic analysis of a sentence are positioned within each other's vicinity in some high-dimensional space. (This is motivated by the topological organization of, for instance, the orientation columns in area V1 in the visual cortex, where neighboring columns are tuned to similar orientations of line segments.)

A syntactic analysis (parse) of a sentence in HPN consists of a trajectory through the network, that (dynamically) binds a set of nodes, as they exchange their topological addresses via a central hub. (This is inspired by research on how primitive visual categories are bound into complex contours or shapes.) By virtue of flexible bindings between the nodes (as opposed to the static bindings in conventional neural networks) HPN can account for the productivity of language.

In HPN, the episodic memory for language is embedded within the semantic memory, in the form of permanent memory traces, which are left behind in the network nodes that were involved in processing a sentence. This way, the network analysis of a processed sentence can always be reconstructed at a later time by means of the memory traces. Moreover, novel sentences can be constructed by combining partial traces of previously processed sentences.

This thesis is organized as follows: Chapter 1 introduces the goals of my research, and motivates the chosen approach. Chapter 2 introduces the Memory Prediction Framework, and provides the neuro-biological background for the neural theory of syntax. Chapter 3 covers some basic concepts from the field of (computational) linguistics, with a special focus on parsing techniques that will be used in the HPN model. Chapter 4 contains a critical review of the literature on neural networks of language processing, within the context of the debate on the fundamental characteristics of structure in language: productivity and hierarchy.

In Chapters 5 to 8 I develop, in multiple stages, the HPN model. In order to quantitatively evaluate the predictions of the neural theory of syntax, I describe a computer implementation of HPN, that allows to run simulations based on tens of thousands of sentences. Chapter 5 starts by introducing the basic HPN model without an episodic memory, which shows HPN's ability to learn a syntactic topology from simple, artificially generated sentences. Subsequently, in Chapters 6 and 7 I discuss an extended model (and computer implementation) that integrates an episodic memory with a semantic memory for language, yet for simplicity lacks a topology. I evaluate this model on a large number of realistic sentences with respect to its performance on syntactic sentence analysis. Finally, in Chapter 8 all the components of HPN are integrated within a single implementation, which demonstrates how an abstract grammar in the form of a network topology is constructed out of episodic linguistic experiences. In this chapter I emphasize the parallels between language acquisition and the process of memory consolidation – the transformation by the brain of information consisting of concrete episodes into a network of abstract, semantic knowledge. In Chapter 9 I present a general discussion and many ideas for future research.

The main conclusion of my dissertation is that it is both possible and worth-

wile to couple insights from (computational) linguistics to neuro-biological insights, and vice versa. On the one hand, from the tough functional demands that language poses on information processing by the brain one can infer a number of non-trivial conclusions regarding neural connectivity and storage in the brain; on the other hand, the physiological limitations of the brain's hardware present some unexpected challenges for theories of syntax, for instance concerning the use of topology.

Titles in the ILLC Dissertation Series:

ILLC DS-2006-01: **Troy Lee**
Kolmogorov complexity and formula size lower bounds

ILLC DS-2006-02: **Nick Bezhanishvili**
Lattices of intermediate and cylindric modal logics

ILLC DS-2006-03: **Clemens Kupke**
Finitary coalgebraic logics

ILLC DS-2006-04: **Robert Špalek**
Quantum Algorithms, Lower Bounds, and Time-Space Tradeoffs

ILLC DS-2006-05: **Aline Honingh**
The Origin and Well-Formedness of Tonal Pitch Structures

ILLC DS-2006-06: **Merlijn Sevenster**
Branches of imperfect information: logic, games, and computation

ILLC DS-2006-07: **Marie Nilsenova**
Rises and Falls. Studies in the Semantics and Pragmatics of Intonation

ILLC DS-2006-08: **Darko Sarenac**
Products of Topological Modal Logics

ILLC DS-2007-01: **Rudi Cilibrasi**
Statistical Inference Through Data Compression

ILLC DS-2007-02: **Neta Spiro**
What contributes to the perception of musical phrases in western classical music?

ILLC DS-2007-03: **Darrin Hindsill**
It's a Process and an Event: Perspectives in Event Semantics

ILLC DS-2007-04: **Katrin Schulz**
Minimal Models in Semantics and Pragmatics: Free Choice, Exhaustivity, and Conditionals

ILLC DS-2007-05: **Yoav Seginer**
Learning Syntactic Structure

ILLC DS-2008-01: **Stephanie Wehner**
Cryptography in a Quantum World

ILLC DS-2008-02: **Fenrong Liu**
Changing for the Better: Preference Dynamics and Agent Diversity

ILLC DS-2008-03: **Olivier Roy**
Thinking before Acting: Intentions, Logic, Rational Choice

ILLC DS-2008-04: **Patrick Girard**
Modal Logic for Belief and Preference Change

ILLC DS-2008-05: **Erik Rietveld**
Unreflective Action: A Philosophical Contribution to Integrative Neuroscience

ILLC DS-2008-06: **Falk Unger**
Noise in Quantum and Classical Computation and Non-locality

ILLC DS-2008-07: **Steven de Rooij**
Minimum Description Length Model Selection: Problems and Extensions

ILLC DS-2008-08: **Fabrice Nauze**
Modality in Typological Perspective

ILLC DS-2008-09: **Floris Roelofsen**
Anaphora Resolved

ILLC DS-2008-10: **Marian Counihan**
Looking for logic in all the wrong places: an investigation of language, literacy and logic in reasoning

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder…": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice

ILLC DS-2011-02: **Fernando Raymundo Velazquez-Quesada**
Small steps in dynamics of information

ILLC DS-2011-03: **Marijn Koolen**
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

ILLC DS-2011-04: **Junte Zhang**
System Evaluation of Archival Description and Access

ILLC DS-2011-05: **Lauri Keskinen**
Characterizing All Models in Infinite Cardinalities

ILLC DS-2011-06: **Rianne Kaptein**
Effective Focused Retrieval by Exploiting Query Context and Document Structure

ILLC DS-2011-07: **Jop Briët**
Grothendieck Inequalities, Nonlocal Games and Optimization

ILLC DS-2011-08: **Stefan Minica**
Dynamic Logic of Questions

ILLC DS-2011-09: **Raul Andres Leal**
Modalities Through the Looking Glass: A study on coalgebraic modal logic and their applications

ILLC DS-2011-10: **Lena Kurzen**
Complexity in Interaction