
Abstract

This thesis introduces GRAEL (grammar evolution) as one of the first research efforts that investigates an agent-based evolutionary computing approach as a possible machine learning method for data-driven grammar optimization and induction. Using the same architecture, but different information sources, GRAEL can be shown to handle a diverse range of grammar engineering tasks, which can help resolve common issues in corpus-based parsing systems, such as insufficient grammar coverage and the suboptimal distribution of probability mass.

After describing a memory-based data-driven parser that applies structure to sentences by direct reference to grammatical information stored in memory, we apply different instantiations of GRAEL to alleviate its aforementioned inherent problematic issues. GRAEL is a distributed system, a computational environment in which agents communicate and co-evolve according to neo-darwinist principles. In the GRAEL environment, each agent is given a partial solution to a problem. By interacting with each other in an evolutionary context, the grammars are optimized in a practical context, on the basis of pre-defined fitness functions.

The first instantiation GRAEL-1 does not alter the content of the corpus-induced grammar and therefore only serves to redistribute the probability mass of the statistical weights of the grammar rules. Experiments show that a careful selection of parameters pertaining to the evolutionary aspects of the environment, can improve performance significantly. The redistributed probability mass can be considered to reflect useful statistics for the task of parsing, rather than mirror the distribution of the original training set.

Grammar-rule discovery (GRAEL-2) can be implemented by allowing the agents in the society to make minor alterations to the rules in the corpus-

induced grammar. The evolutionary computing approach then not only serves as a way to redistribute the probability mass, but also to evaluate the validity of newly created grammar rules. Unsupervised grammar induction can likewise be performed in a GRAEL environment (GRAEL-3) if we allow the agents to build up their own structures, using a minimalist grammar induction approach that employs concepts of information theory to bootstrap structure.

By further reducing the information source, we can leave the engineering aspect behind and transform GRAEL into a computational environment in which we can simulate the emergence of grammatical principles in an artificial language of a group of communicating agents (GRAEL-4). In this view, GRAEL provides a computational simulation of a possible model for the emergence of grammar in early hominids.

Samenvatting

Deze thesis introduceert GRAEL (grammar evolution) als n van de eerste systemen dat een agentgebaseerde aanpak combineert met evolutionair programmeren als een machine learning methode voor corpusgebaseerde grammatica optimalisatie en inductie. Door met dezelfde architectuur verschillende soorten van informatie te verwerken, kan GRAEL een diverse reeks taken uitvoeren met betrekking tot de constructie van grammatica's. GRAEL kan hierbij hulp bieden bij vaak voorkomende problematische aspecten van corpusgebaseerde syntactische analyse, zoals ontoereikende grammaticale dekking en een gebrekkige verdeling van de probabiliteitsmassa over de grammaticale elementen.

In het eerste deel wordt een geheugengebaseerde, corpusgebaseerde parser beschreven die syntactische structuren toekent aan zinnen op basis van eerder geziene grammaticale informatie opgeslagen in het geheugen. Vervolgens gebruiken we deze parser als fundament voor het GRAEL-systeem, waarvan we verschillende implementaties zullen bekijken, die elk een van de voornoemde problematische aspecten van corpusgebaseerde parsers proberen op te lossen. Dit gebeurt aan de hand van een gedistribueerd systeem, een computationele omgeving, waarin agenten met elkaar communiceren en evolueren aan de hand van neo-darwinistische principes. In de GRAEL-omgeving krijgt elke agent initieel een deel van de oplossing wordt aangereikt. Door te interageren met elkaar in een evolutionaire setting, optimaliseren ze deze grammatica's in een praktische context, op basis van vooraf gedefinieerde fitness functies.

De eerste implementatie van de GRAEL-omgeving, GRAEL-1, wijzigt niet de inhoudelijke eigenschappen van de grammaticale informatie die wordt verwerkt. GRAEL-1 probeert enkel de probabiliteitsmassa op een optimale manier te (her)verdelen over de grammaticale elementen in de omgeving. De

experimenten tonen aan dat een zorgvuldige selectie van experimentele parameters die de evolutionaire aspecten van de GRAEL omgeving beregelen, een significante optimalisatie bewerkstelligt van de grammaticale informatie. De dynamiek van de GRAEL-1 omgeving, zorgt ervoor dat de probabiliteitsmassa op een zodanige manier wordt verdeeld, dat de statistische gewichten toegekend aan de grammaticale elementen geoptimaliseerd zijn voor de taak van syntactische analyse zelf.

GRAEL is ook in staat om grammatica's aan te vullen met potentieel nuttige regels. GRAEL-2 implementeert deze functionaliteit door de agenten toe te laten kleine veranderingen toe te brengen aan de regels die worden gebruikt tijdens de communicatie. De evolutionaire aspecten van GRAEL zorgen er dan niet alleen voor dat de probabiliteitsmassa wordt herverdeeld, maar ook dat de nieuwe regels worden gevalueerd in een praktische context, zodat enkel nuttige regels overleven na verloop van tijd. Op een zelfde manier breidt GRAEL-3 de functionaliteit uit naar ongesuperviseerde inductie van volledige grammatica's. De agenten creëren dan uit het niets grammaticale structuren aan de hand van een minimalistisch grammatica-inductie algoritme dat gebruik maakt van entropie. De agentgebaseerde aanpak van GRAEL zorgt er voor dat verschillende alternatieven parallel worden ontwikkeld, terwijl de evolutionaire dynamiek ervoor zorgt dat de beste grammaticale elementen overleven..

Door verder de vooraf gedefinieerde informatiebron af te bouwen, kunnen we GRAEL profileren als een computationele omgeving, waarin we het ontstaan van grammaticale principes kunnen simuleren in de artificiele taal van een groep communicerende agenten. Deze omgeving, GRAEL-4, kan een mogelijke verklaring bieden voor het ontstaan van grammatica in taal.

“There ain't no promise
or guarantee
Ain't no wisdom
To be laid on me
And it's a low down dirty shame
But I gotta find the answer just the same”

Rick Davies - Dead Man's Blues
Slow Motion - ©2002

Acknowledgments

The research presented in this thesis was made possible by the FWO (Fonds voor Wetenschappelijk Onderzoek), who kindly funded me over the course of four years. Their streamlined administration and communication have helped me to minimize the paperwork and devote my time to matters that matter.

I would also like to thank my supervisor Walter Daelemans, who helped me a great deal by applying a sense of direction in my research and who was always there to provide invaluable feedback on whatever subject matter I directed at him, from rough, cretinous ideas, over intermediate results to draft versions of this thesis. I would also like to thank Steven Gillis and Georges De Schutter for reading and commenting on draft chapters. I would also like to take this opportunity to thank the other members of the CNTS whom I have had the pleasure of working with over the years: Gert Durieux, Masja Kempen, Michael Meeuwis (IPRA), Gert Van Rillaer, Erik Tjong Kim Sang, Hanne Kloots, Véronique Hoste, Helena Taelman, Anne Kool, Jakub Zavrel, Eric Van Horenbeeck, Griet Depoorter, Bart Decadt, Evelyn Martens, Marie-Laure Reinberger and Kevin De Coninck. I am especially grateful for their encouraging comments after one of the first presentations of the GRAEL system at CLIN-00, which helped me to resolve my lingering doubts about this line of research. I am also grateful to John Nerbonne, Luc Steels and Remko Scha for agreeing to be in my thesis committee and for taking the

time to read and comment on my thesis thereby providing me with a bird's eye point-of-view on the subject matter.

On the home front, I would like to thank my parents who have always supported me throughout my studies. I would also like to thank Frank for helping me nag to our parents for our first computer, which started my career as a computational linguist. He and Karen are hereby also acknowledged for providing me with a wonderful godchild in Hannelore, who, in 2 years, was able to induce and optimize grammar in a way that none of my agents can. I am also grateful to Sloeber, who tutored me in the ancient art of sustained aestivation. A great deal of gratitude is also due to Tricia, my wife, not only for taking up the ultimately boring task of giving my thesis a native speaker's once-over, but for everything else in my life that really matters. Also a warm thanks to our adopted son and daughter, Banky and Evil, for being noisy, filthy and cute. The obscure inhabitants of the fish tank would also have been acknowledged at this point, but their relevance can not substantiated as such.

It is my belief that you can't undertake the drudgery of writing a thesis without adequate entertainment to counter it. I would hereby also like to thank all the people who made those countless hours of mindless cinematic escapism possible¹. Staying awake after those nightly movie sessions came courtesy of the good people producing the Nalu energy drink. I would also like to thank Ken Stessens, Tom Maes, Gerrit Janssens and Geert De Laet for being partners in crime in our efforts to return rock to its former glory².

Finally, I would like to thank everybody who has helped me in one way or another to get this thesis started, sustained and finished. If for some reason you are not mentioned here, I would like to offer you my apologies for this oversight and my gratitude for your help.

Guy De Pauw
August 2002

¹ Among which: Garland Greene, Truman Burbank, Paul Edgecomb, Jay Phat Buds, Winslow Leach, Corrado Soprano, Jean Vereecken, Castor Troy, Ray Simms, Marty McFly, Corky St. Clair, Nigel Tuffnel, Jack Horner, Biff Tannen, Jiff Ramsey, Jay Billington Bulworth, Chip Douglas, Peter Griffin, Homer Simpson, Dude Lebowski, Will Parker, Mark Borchardt, Michael Corleone, Charlie Baileygates, Paul Sheldon, David Mills, Malcolm Crowe, Dr. Lawrence Jacoby and David Dunn.

² With the help of Sus, Fritz von Kelberg and Knookpeut.

Contents

1	Introduction	1
1.1	Data-Driven Evolutionary Computation	4
1.2	Outline	5
1.3	Navigating the thesis	11
I	Towards A Memory-Based Parser	13
2	A Short Introduction to Memory-Based Learning	15
2.1	Machine Learning	15
2.2	Memory-Based Learning	19
2.3	Natural Language Processing in the MBL-framework	20
2.3.1	Current Research Topics	21
2.3.2	Syntactic Analysis using MBL	22
3	A Memory-Based Approximation of Data-Oriented Parsing	25
3.1	An overview of Data-Oriented Parsing	26
3.1.1	Architecture	27
3.1.2	Experimental Results of DOP	34

3.2	Computational Efficiency of DOP	35
3.3	Pattern-matching: A Memory-Based Approach	37
3.3.1	Memory-Based Parsing	37
3.3.2	Compiling Contextual Information	41
3.4	Experimental Setup	53
3.5	The Parsing Phase	56
3.6	PCFG-experiments	58
3.7	PMPG-experiments	61
3.8	A Combined System (PMPG+PCFG)	62
3.9	Comparative Quantitative Data Analysis	66
3.10	Simple Weighted Voting	73
3.11	An integrated system for PCFG+PMPG	76
3.12	Summary	77
3.13	Current limitations of the research	78
3.14	Conclusions	79

II Data-Driven Experiments in the Evolutionary Computing Paradigm **81**

4	An introduction to GRAEL - GRAMmar EvoLution	83
4.1	Grammar Induction and Optimization - The Engineering Perspective	84
4.1.1	Grammar Sparseness	84
4.1.2	Probabilistic Redistribution	86

4.2	Grammar Induction and Optimization - The Evolution of Language Perspective	90
4.3	Outline of the GRAEL-system	92
4.3.1	Architecture	92
4.3.2	The Parser	97
4.3.3	Different Instantiations of GRAEL	97
4.4	Link with Genetic Algorithms	98
4.5	The Basic GRAEL Algorithms	104
4.6	Concluding Remarks	106
5	GRAEL-1 - An Agent-Based Evolutionary Computing Approach to Probabilistic Grammar Optimization	107
5.1	Experimental Setup	110
5.1.1	General Setup	110
5.1.2	Experimental Parameters	111
5.1.3	Overview of the experiments	114
5.2	Experimental Results: the ATIS corpus	119
5.2.1	20 Agents	120
5.2.2	10 agents	143
5.2.3	5 agents	147
5.2.4	50 agents	151
5.2.5	100 agents	154
5.2.6	General Comments	155
5.2.7	Summary of Results and discussion	158
5.2.8	Extra experiments	160

5.2.9	Some Details about the Data	169
5.3	Experimental Results: the WALL-STREET-JOURNAL Corpus . .	174
5.3.1	20 agents	175
5.3.2	10 agents, 5 agents	180
5.3.3	50 agents, 100 agents	181
5.3.4	The Main Experiment	183
5.4	Advances and Future Work	188
6	A comparison between GRAEL-1 and Ensemble Learning Tech- niques	191
6.1	Bagging	192
6.1.1	Bagging Treebank Parsers	194
6.1.2	Relation to GRAEL	196
6.2	Boosting	197
6.2.1	Boosting Treebank Parsers	198
6.2.2	Relation to GRAEL	200
6.3	Experimental Setup and Results	200
6.3.1	Bagging	202
6.3.2	Boosting	204
6.3.3	“ <i>Bagging</i> ” Agents	206
6.4	Concluding Remarks	207
7	GRAEL-2 - An Agent-Based Evolutionary Computing Approach to Grammar Rule Discovery	209
7.1	Mutation	212
7.2	Experimental Setup	215

7.3	Experiments: ATIS	219
7.3.1	20 agents	220
7.3.2	10, 50 agents	237
7.3.3	Summary of Results and Discussion	239
7.3.4	Extra Experiments	246
7.4	Experiments: Wall-Street-Journal	252
7.5	Advances and concluding remarks	254
8	GRAEL-3: An Agent-Based Evolutionary Computing Approach to Unsupervised Grammar Induction	257
8.1	Unsupervised Grammar Induction	258
8.1.1	Lexical Attraction Modeling	259
8.1.2	Other Distributional Methods	266
8.2	GRAEL-3	268
8.2.1	PS-grammar as a performance model	268
8.2.2	Information Theory for grammar induction	271
8.2.3	Bootstrapping GRAEL-3	283
8.3	Experiments	286
8.3.1	Experimental Setup	287
8.3.2	Restricted Domain - ATIS	288
8.3.3	“Unrestricted” Domain - WSJ	295
8.3.4	Restricted Domain	302
8.4	Advances and Concluding Remarks	306

III	The Emergence of Grammar	309
9	Modeling the emergence of Compositional Language	311
9.1	Computational Simulations of the emergence of syntax: the systems	312
9.1.1	Negotiated Communication	312
9.1.2	Iterated Learning	319
9.1.3	Language Games	324
9.1.4	Other Approaches and Related Research	328
9.2	Computational Simulations of the emergence of syntax: General Tendencies	338
10	GRAEL-4 - Modeling the Emergence of Grammar	343
10.1	Architecture of GRAEL-4	343
10.1.1	The Innateness Discussion	344
10.1.2	The Naming Insight	345
10.1.3	GRAEL-4 Features	348
10.1.4	Communicating GRAEL-4 agents	351
10.2	Experiments with GRAEL-4	364
10.2.1	Setup	364
10.2.2	Experiments: Quantitative view	365
10.2.3	Experiments: Qualitative View	370
10.2.4	Extra experiments	373
10.3	Problematic Issues	374
10.4	Concluding Remarks	377

11 Conclusion	381
11.1 Advances	381
11.2 Future Research	389
11.3 Concluding Remarks	393
A Paths through the chapters	415
B Complete Results for PCFG vs PMPG vs PCFG+PMPG experiments	419
B.1 ATIS	419
B.1.1 Labeled Precision	419
B.1.2 Labeled Recall	420
B.1.3 F-score	420
B.1.4 Exact Match	421
B.2 Wall-Street Journal (10xv)	421
B.2.1 Labeled Precision	421
B.2.2 Labeled Recall	422
B.2.3 F-score	422
B.2.4 Exact Match	423
B.3 Wall-Street Journal (2.21/23)	423
C Correlation between experimental Parameters	425
D GRAEL-1 ATIS Full Results Tables	433
D.1 20 Agents	433
D.2 10 Agents	433

D.3	5 Agents	433
D.4	50 Agents	433
D.5	100 Agents	433
E	GRAEL-1 WSJ Full Results Tables	449
E.1	20 Agents	449
E.2	10 Agents	449
E.3	5 Agents	449
E.4	50 Agents	449
E.5	100 Agents	449
F	124 Unique Rules in GRAEL-2 Test Set	455
G	Grammar Induction failure	457
H	Symbols used in the Penn Treebank	463
I	Set of 100 fixed Meanings	465
I.1	4 Attributes	465
I.2	96 Relations	465
I.2.1	18 x [+ animate] [+ animate]	465
I.2.2	18 x [+ animate] [- animate]	466
I.2.3	60 Complex Relationships	466
J	Abbreviations	469
J.1	GRAEL Instantiations	469
J.2	Terms	470

1

Introduction

“Picassos they’re not. But paintings by a group of pachyderms fetched a pretty penny at a benefit auction at Christie’s on Tuesday night. The acrylic-on-paper paintings drew praise from many of the 300 auction participants. But others snickered at the idea of elephants wielding paintbrushes in their trunks to create modern art. “For thousands of years, elephants have been making mysterious characters on the ground with stones or sticks. Elephant art is only new to people, but it’s not new to the elephants,” said Moscow-born artist Vitaly Komar. The eccentric artist also has exhibited photographs taken by a chimpanzee at an exhibit in Venice, Italy, and aims to work with beavers using processed wooden boards on an architectural project.” [CNN, Art&Style, March 30 2000¹].



Elephants are among a very exclusive club of animals that use tools to

¹<http://www.cnn.com/2000/STYLE/arts/03/22/life.art.reut/>

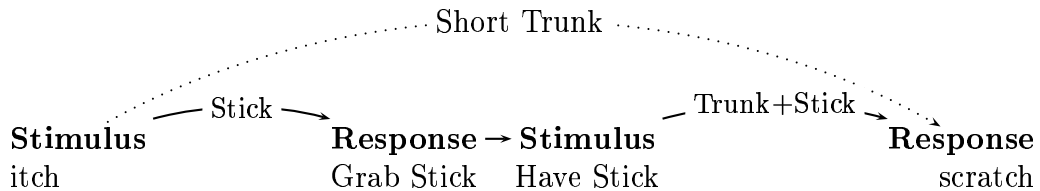
accomplish certain tasks. There are many well documented cases of primates using sticks to obtain food from a narrow hole, or even build primitive constructions that allow them to escape from their compounds. Sea otters and Egyptian vultures have been observed using rocks to break hard shells, while the green heron sometimes uses small bait to catch fish in rivers. And though elephants' trunks mostly wield sticks for less artistic purposes like scratching themselves, the ability to do so inspires people to attribute some kind of higher intelligence to the animals. But why is this?

We usually consider animals as creatures that are very much constrained by their need to comply to basic stimulus-response patterns. What amazes us about animals that use tools to accomplish some kind of task, is that they have found a way to deconstruct a problematic stimulus-response sequence into sub-domains. Without immediately resolving the initial stimulus, the animal defines an intermediate goal that will allow him to reach the response he is after.

An animal may for instance resolve an itch by using one of his limbs, or in the case of an elephant, his trunk, to scratch the area that is bothering him:



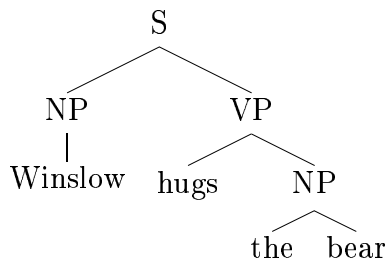
When there is no immediate way to resolve this situation, however, most animals are stumped. An elephant however, can add an intermediate stimulus-response sequence to the solution:



What impresses us about the elephant's behavior at this point is that it no longer appears to be driven by instinctive needs that cause him to observe

the situation in a serial, linear fashion, but that it seems to be a reflective creature, seemingly capable of *analytical* reasoning. As it is wielding its tool, the animal appears to us as having built a structured mental representation of the problem and its solution, which is a cognitive capacity we usually only attribute to humans. Elephants, otters, apes and the like seem to be able to apply structure to a situation and mentally construct a sequence of rules, i.e. a *grammar* that can help them to solve problems.

Many researchers indeed suggest that it is exactly this capacity that was paramount to the development of linguistic abilities in early hominids. The cognitive capacity to construct a structured mental representation of the world that surrounds us, has indeed not only allowed us to manipulate it with man-made tools, but also reflect on it and communicate about it as well. It is therefore not surprising that the analytical nature of our thoughts is reflected in its external manifestation as well: language.



When we communicate a sentence like “*Winslow hugs the bear*”, we provide a linear string of tokens to our listener. But this linear string is the by-product of a whole set of principles, which ensures for instance that the words are placed in the right order. We can hypothesize that these principles cause us to keep some kind of syntactic structure in mind for the sentences we speak, much in

the same way an elephant has built a structured mental representation of the problematic situation of scratching, which in essence constitutes a grammar that describes the problem and its solution (Figure 1.1).

Whether this structured representation is a mental reality is another matter entirely and one we will do our best to avoid. As hard as it is to imagine that elephants have induced grammars for scratching, it is equally challenging to put forward likewise claims for the syntactic representations used by a language user. All we can say is that the externalized behavior can a posteriori be structured using the type of representation featured in Figure 1.1. The research described in the following chapters will try to implement computational methods that construct syntactic representations for language utterances, drawing from a variety of information sources and using a number of different techniques.

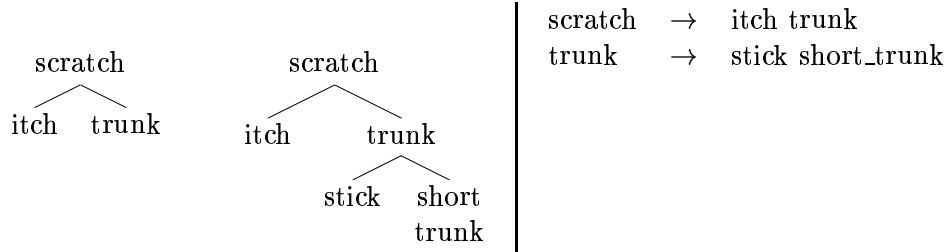


Figure 1.1: A treebank and grammar for the problem of “Scratching with a (short) trunk”

1.1 Data-Driven Evolutionary Computation

After developing a computational system for syntactic analysis that uses a corpus of examples as its information source, we will describe a set of experiments that tries to improve grammars by introducing them into a dynamic system using **evolutionary computing** for problem solving. Evolutionary computing, a technique dating back to the late 1950’s [Box 1957; Fraser 1957; Bledsoe 1961], tries to implement computational models that employ concepts drawn from biological evolution and is a cover term for a whole range of methods, such as genetic algorithms and programming, classifier systems and evolutionary programming. [Spears et al. 1993] define evolutionary algorithms as follows:

[Evolutionary algorithms] maintain a population of structures, that evolve according to rules of selection and other operators, that are referred to as “search operators”, such as recombination and mutation. Each individual in the population receives a measure of its fitness in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. Although simplistic from a biologist’s viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

So rather than focusing on mimicking biological evolution itself, evolutionary algorithms try to employ these principles as a search algorithm throughout a search space of possible solutions. The GRAEL-environment² we will describe, establishes a society of individuals that each hold a grammar. By interacting with each other in an extended series of communicative attempts, the grammatical information may be **recombined** and **mutated**, thereby yielding new grammars. By defining a set of **fitness** functions, we can **select** which individuals hold the kind of grammar that we are after and consequently optimize them for some particular task over time.

In the GRAEL-environment, we combine the concepts of evolutionary computing with an agent-based approach. By incorporating the information to be optimized in an autonomous agent, we provide a method to fine-tune the selection process: by defining a number of fitness functions based on inter-agent communication, we can select grammars that have been optimized in a practical context, i.e. during actual **parsing**, rather than hypothesize over their quality in terms of their formal or distributional properties.

The GRAEL environment provides a **distributed** approach to grammar optimization, meaning that a variety of different alternatives are considered and developed over time. Since there is a large random factor at play in these types of experiments, this approach therefore also provides a way to resolve the problem of local maxima, i.e. finding a grammar that is the best overall solution, rather than a reasonable intermediate solution.

The experiments described in Part II of this dissertation will show that corpus-induced grammars can indeed be optimized and induced by applying a distributed/agent-based evolutionary computing method to the problem, while Part III will describe experiments that apply a similar approach to model the emergence of grammatical language in a computational context.

1.2 Outline

We can delineate three major parts to this dissertation, each trying to provide a computational description of the structural properties that can be found in natural language. In Part I we implement a system that can provide a

²Short for **GRAM**mar **EvoL**ution.

syntactic structure for a sentence on the basis of examples. For this purpose, we use a large collection of sentences that have been provided with tree-structures by human annotators. The parsing system we describe is able to mimic the annotators' behavior by inducing statistical and grammatical information from these tree-structures. Part I therefore describes the most straightforward experiment: the implementation of a memory-based parsing system that is able to provide syntactic structures for sentences.

This approach is however by definition restricted by the material we have to work with, i.e. the annotated corpus. Information may be lacking from the corpus, or the information may be distributed in a way that is not optimized for the task of providing a syntactic structure itself. In Part II we implement an **agent-based evolutionary computing** approach to overcome these restrictions. While the distributed nature of the system makes sure that a wide range of alternatives can be considered simultaneously, the evolutionary computing aspects provide a way to distinguish the good from the bad over time. This method will eventually allow us to devise a grammar induction system that goes a long way in providing syntactic structures without the need for pre-annotated examples.

Parts I and II describe systems that can be integrated in practical applications for natural language understanding. Part III on the other hand, will use the sensibilities of the distributed evolutionary computing approach, to model the emergence of grammatical language in a computational context. We will try to show that grammar can emerge in a population of agents, to which we attribute a minimal amount of presupposed cognitive abilities.

Despite the fact that the three parts differ from each other in the aspect of computational syntax they try to elucidate, there is a continuous thread running through the entire dissertation that binds all chapters together. We start off with a parsing system in Part I that tries to provide analyses for natural language sentences using pre-defined grammatical structures and eventually end up in Part III with a computational model that tries to create some artificial compositional language. Part II picks up on some of the systems that lie between these two extremes. The recurrent theme that binds the different systems together is therefore a deconstructionist one: starting off with a fully specified parser, we gradually abandon the pre-defined information sources that are available and end up with a minimalist investigation

into the dynamics of grammar in a computational context.
















Table 1.1 displays this development. Chapter 3 starts off with a system that is fully specified in terms of the information sources that are available for processing. Using concepts from the machine learning field of Memory-Based Learning (introduced in Chapter 2), we redefine a popular approach to corpus-induced parsing, called Data-Oriented Parsing. The Pattern-Matching Probabilistic Grammar (PMPG in Table 1.1), uses a corpus of annotated tree-structures to commit large chunks of syntactic contextual information to memory. The PMPG is consequently able to provide syntactic structures for new, previously unseen sentences, by retrieving the structures recorded in memory and combining them into a new structure, using probabilistic information also extracted from the corpus.

Table 1.1 indicates that we consider the PMPG's information source to be fully specified: it uses an existing **language**, i.e. English, and a pre-defined **grammar**, induced from an annotated corpus. The grammar describes how words in a sentence are grouped together and how those *constituents* are consequently combined into higher-order structures. We state that the **segmentation** properties³ are also predefined: at no point during the processing, does the parser consider deviations from the set of segmentation rules provided by the grammar. Finally, the **probability mass** that is used to find the best combination of constituents is also set, as it is directly induced from the annotated corpus.

In the discussion of the fully specified parsing system from Chapter 3, we will identify a number of inherent limitations to the system. A lot of erroneous structures are generated that are caused by (i) a suboptimal distribution of the probability mass and (ii) insufficient grammatical information in the annotated corpus. We therefore set out to find a generalized grammar induction and optimization technique. Chapter 4 describes this approach, using an agent-based evolutionary computing environment called GRAEL. By distributing grammatical knowledge over a society of agents and having them interact with each other in an extended series of communicative attempts, grammatical information is optimized in an evolutionary context. The agent-based approach allows a number of alternatives to be developed

³Segmentation properties describe the way in which words are grouped into segments, i.e. constituents.

Table 1.1: A deconstructionist Setup

	Utterances	Natural Language	Grammar	Segmentation	Probability Mass
PMPG					
GRAEL-1					
GRAEL-2					
GRAEL-3					
GRAEL-4					

simultaneously, while the evolutionary computing aspects of GRAEL ensure that the impact of marginal grammatical structures is diminished over time.

Chapter 5 describes the GRAEL-1 system which abandons the pre-defined distribution of the **probability mass**, as featured in the PMPG-system. In GRAEL-1 grammatical structures extracted from the annotated corpus are distributed over a society of agents. This provides each of the agents with a basic initial grammar for parsing. The agents will then start to practice their grammars on each other. Agents help each other out by explicitly providing structures to each other that are relevant for obtaining the correct structure. As the agents are exposed to new chunks of grammatical information, they re-adjust the probability mass in their grammar in a context that reflects the actual task at hand: parsing new sentences. Rather than mirroring the distribution in the original annotated corpus, the probability mass in the agents' grammar accommodates useful statistics for parsing. GRAEL-1 only changes the distribution of the probability mass, while the other information sources remain intact: segmentation properties are being respected by the agents and in fact none of the grammatical properties in the annotated corpus are altered. As a matter of course, the actual language remains constant as well.

Whereas GRAEL-1 addresses the problematic distribution of probability mass in data-driven grammars, GRAEL-2 tries to solve the issue of *grammar sparseness*. We will often find that the syntactic structure for a previously unseen sentence requires a rule that is not induced from the annotated corpus. Not even the optimized probabilities that GRAEL-1 provides can resolve this situation. As the rules that are needed are often not more than a variant of some existing rule, we abandon the pre-defined **segmentation** properties provided by the annotated corpus in GRAEL-2. Chapter 7 describes how GRAEL-2 expands the functionality of the GRAEL environment to grammar rule discovery by applying *mutation* on existing grammar rules. The distributed nature of the GRAEL environment again makes sure that enough alternatives are being considered, while the newly created structures are being practiced on and evaluated in a practical context. The evolutionary aspects of GRAEL consequently make sure that useful rules are distinguished from useless ones.

Whereas the grammatical structures in GRAEL-2 were still largely based on those provided by the annotated corpus, we abandon any pre-defined

grammatical information source in GRAEL-3 (Chapter 8). This effectively turns the GRAEL environment into a fully unsupervised grammar induction technique. A simple grammar induction method based on concepts from information theory bootstraps structure in the society, after which the society returns to business as usual: each agent develops an alternative grammar, while the interaction between agents and the evolutionary approach makes sure that the grammars are optimized over time. The only information source that is still pre-defined in GRAEL-3 is the language itself: agents in the society are provided with a number of sentences, the structure of which is implicitly present in the distributional properties of the words. This relates to the discussion of the mental representation of syntactic structure. Whereas GRAEL-1 and GRAEL-2 allowed agents to explicitly store grammatical structures in their memory, GRAEL-3 abandons this view in favor of a less presumptuous stance that considers syntactic structure as a mere representation of the externalized realizations of the language capacity, rather than a mental reality.

This view allows us to consider a computational model that studies the emergence of grammar in a society of agents. By providing the agents with a minimal linguistic capacity, we can model the emergence of compositional language solely on the basis of externalized properties of language and principles of co-evolution. After looking at some alternative approaches in Chapter 9, we will develop such a system in Chapter 10: GRAEL-4. In a GRAEL-4 system we abandon the last pre-defined information source that is left: the language. All that is left are random utterances, in which initially grammatical structures are implicitly present. The agents will try to provide some basic structural representation of the sentences they observe, which may eventually enable them to guide the construction of their own sentences. As the society goes from grammatical chaos to a general set of grammatical principles that the agents mostly adhere to, a grammatical language emerges in the society.

Starting out with a fully specified parsing method in Part I, we have gradually abandoned every pre-defined information source initially available to the parser in the different GRAEL instantiations in Part II. The end-result is described in Part III and comprises of a system that creates a basic compositional language out of a society of agents that initially only were able to produce unstructured utterances.

With GRAEL, we hope to present one of the first research efforts that introduces agent-based evolutionary computing as a machine learning method for data-driven grammar optimization and induction. The GRAEL environment can then more generally be considered as a framework that allows for the simultaneous development of a range of alternative solutions to a grammatical problem, optimized in a series of practical interactions in an environment controlled by evolutionary parameters.

1.3 Navigating the thesis

Given the variety of tasks that we set out to achieve, not all chapters may be equally relevant to every reader. Appendix A can be consulted to define the reader's path through the chapters. Generally, researchers that are partial to engineering problems, may prefer to read Chapters 3 to 8, while researchers in the field of evolutionary computing will rather consult Parts II and III than Part I. Formal linguists may find pointers for discussion in the first sections of Chapters 3 and 8 and Chapters 9 and 10.

We hope however that most readers will follow at least the recommended path outlined in Appendix A, as it will provide a general overview of how memory-based syntactic parsing can be combined with a distributed evolutionary computing approach to tackle several different grammar optimization and induction tasks. It speaks for itself however that the only way to grasp a full understanding of all aspects of the experiments, is to simply read the text in its entirety.

Appendix J contains a list of abbreviations that are used throughout this thesis. We recommend that the reader peruses this list to make abbreviations, most often used in results tables and the like, more transparent.

We now turn to the first part, in which we describe the parsing backbone for the main experiments in Part II, as well as introduce some concepts paramount to syntactic parsing in a computational context.



Part I

Towards A Memory-Based Parser

Memory can change the shape of a room; it can change the color of a car. And memories can be distorted. They're just an interpretation, they're not a record, and they're irrelevant if you have the facts.

Leonard Shelby - Memento - ©2000

2

A Short Introduction to Memory-Based Learning

2.1 Machine Learning

Researchers in the field of Machine Learning try to implement algorithms that allow computers to improve their performance on a particular task, by *learning* from experience, rather than employing pre-programmed expertise. In contrast to *expert systems* (i.e. computational implementations of a domain expert's knowledge) machine learning algorithms are portable in that they can be trained to perform a new task without having to recode the algorithm itself. One simply needs to present adequate data about the domain at hand to the machine learner in order to turn it into an "expert" for that domain. This data may need to be annotated, which still requires human intervention, but in theory, machine learning algorithms should be able to discern useful information from useless information in the data, thereby eliminating the need for a human domain expert and consequently the knowledge acquisition bottleneck that engineering expert systems entails.

It is therefore not surprising that the machine learning paradigm has become a major player in the field of computer science, bringing together researchers from the fields of Artificial Intelligence, statistical processing, logic programming, natural language processing, biology and the like. While the engineers have kept busy refining the machine learning algorithms to make them more powerful, more efficient and more portable, the end-user researchers have employed the methods to gain new insights into their subject matter, as well as use the systems for practical problem-solving or classification tasks.

Machine learning algorithms hold in common that they *learn* from experience, i.e. examples, in some way or another. These examples are called the *training data*. By processing the training data, the learner will be able to perform a particular task. These tasks can most often be described as *classification tasks* or *disambiguation tasks*. Given an item with a particular set of properties, a machine learning algorithm needs to choose a solution from a finite number of possibilities. The machine learner employs some kind of processing on the *training data* it has been presented with, to model the information needed to trigger the correct disambiguation. To test whether the system has actually learned anything, an independent *test set* is used for evaluation purposes.

The concept of *learning* itself is controversial: can a memory-based system, which stores information in memory for instance, be considered as having learned anything? The most frequently adopted stance in this matter is to view the system as somewhat of a black box: if it can provide a solution for a problem on the basis of data provided to the systems, it is considered as having learned to solve the problem at hand.

Most often the data in Machine Learning experiments is presented by a (usually large) number of feature values. Each *instance* in the data consists of a number of **features** and is associated with a particular **value** or **class**. The features describe the properties of the instance, while the value denotes the correct classification of the instance.

The data in the next example (taken from [Quinlan 1993]) models whether or not the weather is suited for playing outside. Each instance represents a particular weather situation, with features such as the temperature and the humidity. To each instance of this training data, a class (play/don't play)

has been assigned:

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	CLASS
sunny	75	70	true	Play
sunny	80	90	true	Don't play
overcast	72	90	true	Play
overcast	83	78	false	Play
rain	65	70	true	Don't play
rain	70	96	false	Play

By processing this data, the Machine learner will learn to classify new data. Given a meteorological situation such as

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	CLASS
rain	83	96	true	?

the system may choose to classify this *instance* as **don't play**, perhaps because it has learned that a high value for HUMIDITY, combined with strong winds and a high temperature constitute an unsuitable condition for playing outside.

Different machine learning algorithms will provide different classification decisions. In other words: each machine learning algorithm brings a particular **bias** to a system [Breiman 1996; Mooney 1996; Roth 1998; De Pauw and Daelemans 2000]. A machine learning algorithm will need to resolve issues such as the relevance of each feature (e.g. is the feature WINDY relevant towards classification), interdependence of features (e.g. is the feature HUMIDITY dependent on the feature RAIN) and combinatory effects of features (e.g. does a high humidity combined with a low temperature trigger a particular kind of classification), ... All of these issues constitute a search space through which a machine learning algorithm has to traverse to find the correct solution. An algorithm introduces bias into the system in the way it employs different search heuristics to organize its search.

[Langley 1996] describes five paradigms for machine learning, to which we would like to add one more for convenience's sake:

1. **Neural Networks** Neural networks transform data into a network of interconnected layers of nodes. In the training phase, the data adjusts the weights of the links in that network. This network can subsequently be used to classify new instances: a particular input will activate the relevant nodes from the input layer up to the output layer, the latter of which indicates the classification for the instance.

2. **Rule Induction** This method transforms the training data into decision rules or a decision tree. New data is classified by applying the rules/tree to the instances.

3. **Instance-Based Learning** The learner tries to match new instances by direct reference to the training data and does not transform the instances into some new representation, such as a network or decision tree.

4. **Genetic Algorithms** A method inspired by biology, in which typically (sequentially ordered) rules are recombined into stronger classifiers by using a neo-Darwinist survival-of-the-fittest type evaluation method.

5. **Analytic Learning** Uses logical form rules to construct proofs which are consequently combined to construct new complex rules that solve similar problems more efficiently.

6. Statistical Methods

Can also be regarded as a machine learning method. New data is classified by considering the probabilistic distribution of the training data.

Method 1, 2 and 5 will not be dealt with in this dissertation, but we will deal with statistical methods in Part I and more generally with information theory concepts in Parts II and III. The concepts of genetic algorithms and the entire field of evolutionary computing in general are the basis for the GRAEL-system described in Parts II and III and Memory-Based Learning is the basic machine learning algorithm underlying this dissertation.

While most of these systems can be implemented either way, one should make a distinction between *batch learning* and *incremental learning*: the former processes the training data in one sweep, while the latter allows new training data to be added to the system along the way. Incremental learning therefore constitutes a psychologically more realistic learning method.

One last distinction needs to be pointed out: supervised vs unsupervised learning. Most of the machine learning research is involved in *supervised learning*, in which a training instance is associated with a particular class. This provides the system with explicit classification examples. *Unsupervised learning* on the other hand does not provide the correct classification for the training instances. The machine learning algorithm will have to look at the distributional aspects of the instances on a global level for instance, to deduce proper classification/problem-solving behavior. Chapters 3 to 7 will deal with supervised methods, while Chapters 8 and 10 will describe an unsupervised learning method.

2.2 Memory-Based Learning

One of the most straightforward methods in the field of machine learning is *Memory Based Learning*, an instance-based learning method [Aha et al. 1991; Kolodner 1993]. Memory-Based learning assumes that problems can be solved by direct reference to similar, previously observed events. This seems relevant for many classification tasks such as the aforementioned weather forecast example: one might look at the most similar weather situation that

was previously observed and extrapolate the decision whether or not to play outside to the current situation.

Memory-Based Learning is a *lazy learning* approach in that it does not process the data in any way [Aha 1997]. Even though data is typically presented as feature values, which implies some pre-processing of the data, memory-based learning does not alter the properties of the data itself by transforming it into a different representation (e.g. neural nets, decision trees), nor is the probabilistic distribution of the data computed during or after the knowledge acquisition phase. Memory Based Learning models the acquisition of knowledge through simple storage of events in memory without further processing.

When a new instance needs to be classified, the most similar items in memory (the *k-nearest neighbors (k-nn)* [Cover and Hart 1967]) are determined and their classification extrapolated. The most important problem in MBL is exactly how this *similarity* is computed. A simple method counts for each instance in memory, the number of features it holds in common with the instance to be classified.

Distance metrics can be introduced to determine the relevance of the features toward classification. Determining the nearest neighbor of an instance then does not only involve counting the number of features they hold in common, but also the weights associated with each feature. There are a wide range of extensions to the MBL method, many of which are outlined in [Daelemans et al. 2001], including IB1-IG [Daelemans and Van den Bosch 1992] and the IGTREE-method [Daelemans et al. 1997], an efficient approximation of Memory-Based Learning that compresses the data in a decision-tree type structure for efficient disambiguation.

2.3 Natural Language Processing in the MBL-framework

Given the psycholinguistic relevance of MBL [Skousen 1989; Chandler 1992; Gillis and Durieux 2000], it should come as no surprise that it has been widely applied to a wide range of linguistic classification tasks as well [Daelemans 1999; Roth 1999], such as part-of-speech tagging, grapheme to phoneme con-

version and stress acquisition.

The description of natural language is by definition problematic, due to the many exceptions and irregularities that are inherent to the domain [Daelemans et al. 1999]. Approaches that try to model natural language phenomena by inducing regularities from the data are often not robust enough to discern exceptions from noisy data. The MBL approach is therefore more suited: exceptions in the data are simply stored in memory for future reference and will only come into play when the general tendencies in the data are not sufficient to trigger the correct disambiguation.

2.3.1 Current Research Topics

An exhaustive overview of the field of Memory Based Language Processing can be found in [Daelemans et al. 2001]. Most of the publications listed there originate from the research groups ILK (KUB, Tilburg, The Netherlands) and CNTS (UA, Antwerp, Belgium). Typically, this line of memory-based learning research focuses on using a *propositional* representation of the problem at hand. Most often the data is presented as feature values, such as the one in the play/don't play classification task. TIMBL [Daelemans et al. 1998] provides a useful tool for memory-based natural language processing. It incorporates a number of different implementations and optimizations of the memory-based learning algorithm and can deal with symbolic, as well as numerical features.

Even though the distinction is usually not explicitly made, we can in general distinguish two types of research efforts: those that concentrate on the engineering aspect of processing natural language in a memory-based framework (e.g. [Daelemans et al. 2000], [Zavrel and Daelemans 1997] and [De Pauw and Daelemans 2000]), while other papers detail the linguistic aspects of these implementations (e.g. [Gillis and Durieux 2000], [Daelemans et al. 1997] and [De Pauw 2000a]).

A large part of current machine learning research involving memory-based learning is also dealing with combinatorial methods, such as bagging [Breiman 1996; Hoste and Daelemans 2000], boosting [Hoste and Daelemans 2000; Schapire 1999; Abney et al. 1999; Henderson and Brill 2000] and system combination [van Halteren and Daelemans 2001]. The combination of clas-

sifiers is able to improve on disambiguation accuracy by re-distributing data or combining the sensibilities of machine learning methods. Chapter 3 will deal with system combination as well and Chapter 6 will provide a comparison between the distributed evolutionary computing model described in this dissertation and the bagging and boosting methods.

2.3.2 Syntactic Analysis using MBL

Since this dissertation tries to implement syntactic analysis in a memory-based framework, we will briefly overview related research efforts in this field. Section 2.3.1 defined three major subfields for memory-based syntactic processing: *shallow parsing*, *full parsing* and the retrieval of *grammatical relations*.

Shallow Parsing

Often denoted as *chunking*, shallow parsing constitutes the task of finding shallow syntactic patterns and relationships in a sentence. It is unlike normal parsing, in that it does not build a full syntactic representation of the entire sentence, but only delineates particular syntactic clauses and assigns a grammatical category to them. Shallow Parsing typically combines two tasks: (i) a segmentation task (i.e. finding constituent boundaries) and (ii) a disambiguation task (i.e. labeling constituents, finding grammatical relations, ...). Dividing the parsing problem into sub-domains makes it possible to use propositional feature values representations. Common syntactic parsing pitfalls, such as long-distance dependencies and recursion, are therefore dealt with in specialized modules.

Many machine-learning algorithms have been applied to this problem: [Ramshaw and Marcus 1995] use Transformation-Based Learning, [Vilain and Day 1996] employ rule sequences, [Argamon et al. 1998; Daelemans et al. 1999; Veenstra 1998; Buchholz and Daelemans 2001] implement memory-based shallow parsing and [Cardie and Pierce 1998] use a similarity-based method.

[Daelemans et al. 1999] reports on a memory-based shallow parsing

method, in which chunking is described as a tagging task. Each word in the sentence needs to be attributed one of five tags: **I_NP**(word inside a baseNP), **I_VP**(word inside a baseVP), **O**(outside a baseNP or baseVP), **B_NP**(word inside a baseNP but the previous word belongs to another baseNP) and **B_VP**(word inside a baseVP but the previous word belongs to another baseVP). This approach is able to achieve accuracy scores that compare favorable to other approaches. [Daelemans et al. 1999] also demonstrates the modular approach to shallow parsing, by describing an extension that detects subject/object relations, using the output of the chunker.

Combining a competitive classification accuracy with attractive computational complexity, propositional machine learners for shallow parsing therefore have several advantages over full parsers for practical applications. Furthermore, for most applications, full syntactic analysis is not needed. Information retrieval systems may already benefit from knowing the content of the semantically highly informative NP-units, so that only an NP-chunker [Veenstra 1998] would suffice. A complete tree-structure of the sentences in the documents would not add much to the accuracy of these systems and would only introduce computational overhead.

Finding Grammatical Relations between constituents

Often considered part of shallow parsing, the task of finding grammatical relations between constituents is complicated enough to warrant a separate discussion. One of the most common examples of this task is the PP-attachment problem. [Zavrel et al. 1997] describes a memory-based approach to this problem which compares favorably to statistical approaches [Franz 1996] and other machine learning algorithms [Brill and Resnik 1994; Ratnaparkhi et al. 1994].

More closely tied to shallow parsing is research that automatically tries to model sub-categorization properties [Buchholz 1998] and cascaded grammatical relations assignment [S. Buchholz 1999], the latter of which provides a link to full parsing.

Full Parsing

On a general level, we can define two types of full parsing. First, **cascaded shallow parsing**, described in [Daelemans et al. 1999; Tjong Kim Sang 2001; S. Buchholz 1999] tries to build full parsers by cascading memory-based models that deal with subproblems of parsing, such as chunking (segmentation) and the retrieval of grammatical relations (labeling). The advantage of using such a modular approach to parsing is that, as opposed to full parsing, the segmentation and labeling task can be broken down into subproblems. Sentences can be provided with a structure, using a cascaded model of classifiers trained on propositional representations for these subproblems. [Tjong Kim Sang 2001; Tjong Kim Sang 2002] shows that a chunker can be transformed into a bottom-up parser by recursively applying it to its own output. Although it is outperformed by the current state-of-the-art full parsing systems, it offers an interesting method for full syntactic parsing, using only propositional feature value representations.

Not a lot of research has been done to apply the insights of memory-based learning to **full syntactic analysis**. [Scha 1999; Bod 1998] describe Data-Oriented Parsing, the memory-based aspects of which have been made more explicit in [De Pauw 2000a], while MBSL [Argamon et al. 1998] combines the two different approaches to memory-based sentence analysis. It combines the Data-Oriented Parsing method of considering all substrings of a sentence with the modular classification-based approach of shallow parsing as described in [Daelemans et al. 1999].

In this chapter, we have introduced some basic concepts of machine learning and memory-based learning in particular, which will be the keystone to subsequent chapters. In Chapter 3 we will describe a parsing system for natural language that establishes a more explicit interpretation of memory-based learning concepts in Data-Oriented Parsing. This memory-based parsing system will provide the grammatical backbone for most of the experiments in Part II.

All life is pattern, but we can't always see the pattern
when we're part of it.

Belva Plain
Crescent City - Delacorte 84

3

A Memory-Based Approximation of Data-Oriented Parsing

As we discussed in Chapter 2, the current practice in Memory Based Language Processing is to encode the linguistic information needed to trigger the correct solution, in a propositional format (*feature values*) and present it to a memory based learner, such as TIMBL [Daelemans et al. 2001]. Yet, many of the intricacies of the domain of syntax do not translate well to this kind of propositional representation, so that established MBL-methods for syntactic analysis are necessarily geared to models that cascade different levels of low-level syntactic analysis. With its emphasis on memory as the main syntactic knowledge base, Data Oriented Parsing however provides an opportunity to consider a Memory Based model for full syntactic analysis, that avoids dividing the problem into sub-domains.

But although Data-Oriented Parsing (henceforth DOP) can be considered as a memory-based approach to syntactic analysis, some key issues of MBL are only implicitly employed. This chapter describes a re-interpretation of the DOP-model, in which the memory-based aspects of the model are exploited,

so that any given parse is evaluated in terms of its similarity to previously recorded syntactic analyses in memory, in the same vein as established MBL-techniques. Furthermore, this memory-based approach also resolves some of the computational efficiency issues that are inherent to the prototypical DOP-methodology.

This chapter will start off with an introduction of DOP in Section 3.1, after which Section 3.2 will discuss the computational aspects surrounding DOP. Section 3.3 will introduce some MBL-terminology in the context of this chapter. Section 3.4 describes the experimental setup and the corpora used for these experiments. The parsing phase that precedes the disambiguation phase will be outlined in Section 3.5 and a description of the three disambiguating models can be respectively found in Sections 3.6, 3.7 and 3.8. After the data-analysis in Section 3.9, an integrated model is discussed in Section 3.11, after which we conclude by summarizing the models described in this chapter and addressing some current limitations to the research.

3.1 An overview of Data-Oriented Parsing

Data Oriented Parsing, originally conceived by [Scha 1990] and applied to natural language parsing by Rens Bod (consult [Bod 1998] for an overview) translates the psycholinguistic insight that language users analyze sentences using previously registered **constructions** and that not only simple rewrite rules, but also complete syntactic substructures of arbitrary size can be linguistically relevant units for parsing [Fenk-Oczlon 1989; D. Mitchell and Corley 1992; Jacoby and Brooks 1994]¹.

Classic methods for syntactic analysis employ a context-free grammar, typically consisting of a large number of rewrite-rules, to power a parser which is able to generate a *parse forest* for a sentence. A parse forest consists of all the possible syntactic parse trees for a sentence that are consistent with the grammar. But whereas a parser will propose many different analyses for a sentence, language users typically only perceive one analysis. [Bod 1998] argues that a parser only using rewrite rules may therefore constitute a *competence* model, but not a *performance* model in that it does not prefer

¹References reproduced from [Bod 1995].

any particular parse over the other.

A statistical parsing method is able to express this kind of preference by employing statistical information in the grammar. A PCFG² for example is a normal context-free grammar in which each rewrite rule has some kind of probability attached to it. To compute the probability of a tree-structure (i.e. its “preferability”), one can simply multiply the probabilities of the rewrite-rules that were used to construct the parse. The probabilistic values attributed to the rules of a PCFG can be easily induced from an annotated corpus.

Data Oriented Parsing is similar to such a PCFG-method in that it also uses corpus-induced statistics to disambiguate a parse forest. But the DOP-approach is novel as it also induces the grammatical information itself from an annotated corpus in the form of fragments of tree-structures. By not imposing any limit on the depth of the grammatical elements³, DOP introduces context-sensitivity, capturing dependencies and idiomatic constructs in a way that a simple CFG is unable to do.

3.1.1 Architecture

[Bod 1998] defines four pre-requisites for a DOP system:

- a corpus of annotated tree-structures
- the substructures of these tree-structures
- a combination operation to combine these substructures
- a combination operation to combine the probabilities of these substructures

The core of a DOP-system is its *treebank*: an annotated corpus is used to induce all substructures of arbitrary depth, together with their respective probabilities, which is expressed by its frequency in the treebank relative

²Probabilistic Context-Free Grammar

³The depth of a grammatical element in a CFG being 1.

to the number of substructures with the same root-node. In principle, it should not matter what flavor of (ps-)grammar formalism has been used to construct the trees, as the DOP method can be extrapolated to process any kind of tree-based representation. The actual psycholinguistic relevance of using ps-grammars as a formalism for representing syntactic structures, is therefore not a key issue in DOP experiments. DOP is typically applied to phrase structures, like the ones found in the Penn Treebank [Marcus et al. 1994].

A toy example of an annotated corpus is displayed in Figure 3.1. To construct the treebank which powers the DOP model, we must now extract every single fragment of each tree-structure. The resulting treebank can be found in Figures 3.2 and 3.3. Notice that the full tree-structure is included in the treebank, as well as the smallest possible structures of depth 1. These are equivalent to simple rewrite rules:

$$\begin{array}{c} \text{VP} \\ \swarrow \quad \searrow \\ \text{offered} \quad \text{NP} \end{array} \quad = \quad \text{VP} \rightarrow \text{offered NP}$$

Using this treebank, new, previously unseen syntactic structures and sentences can be generated/parsed, such as *Brian offered some bear a haircut*. Whereas a CFG would have only one way of forming this structure (namely the one described in Figure 3.5), a treebank of tree-structure fragments of arbitrary size such as the one DOP uses, has different possibilities in generating the same structure. In other words, the tree-structure for *Brian offered some bear a haircut* has multiple **derivations**, among which those featured in Figures 3.4 and 3.5.

To compute the probability of a parse-tree, we need to look at the probabilities of the substructures that were used to construct it. Referring back to the treebank in Figures 3.2 and 3.3, we calculate for each substructure its relative frequency, i.e. the number of times it occurs in the treebank and divide it by the number of times a substructure with the same root-node occurs. For example:

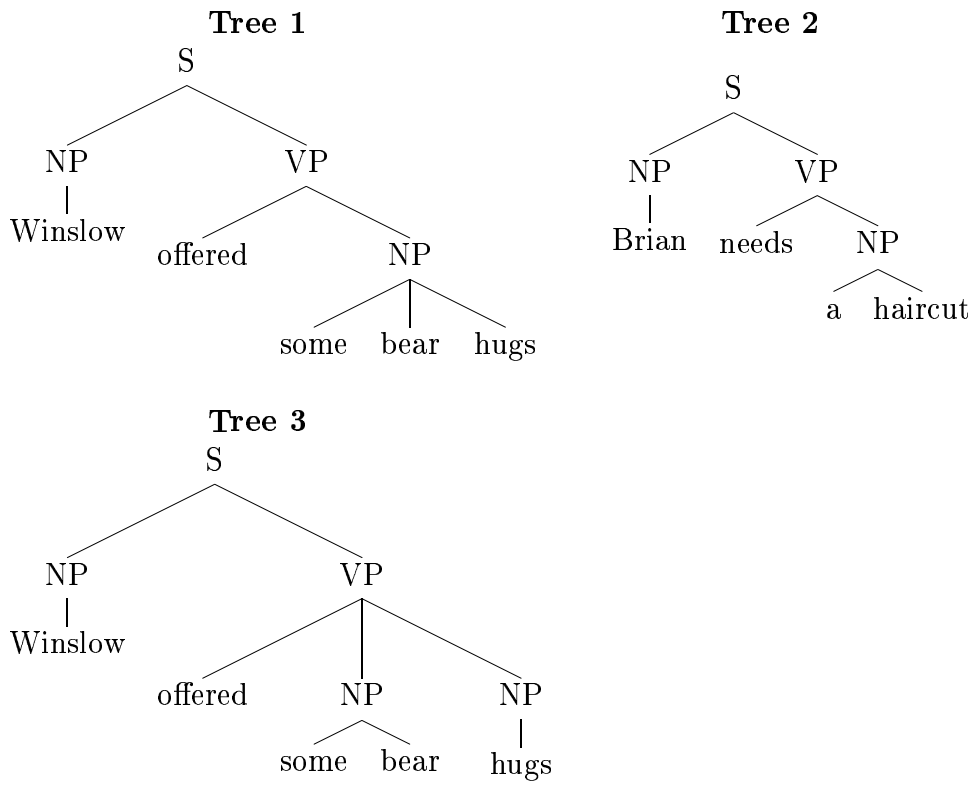
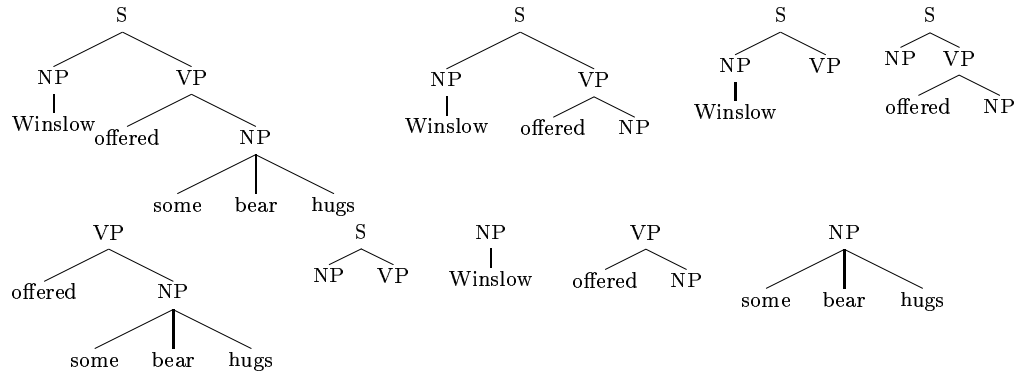


Figure 3.1: A Toy Corpus for DOP

Tree 1



Tree 2

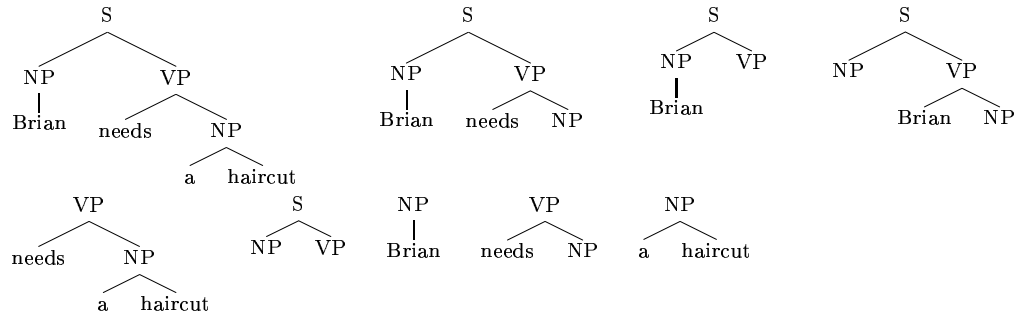


Figure 3.2: Treebank for Tree 1 and 2

Tree 3

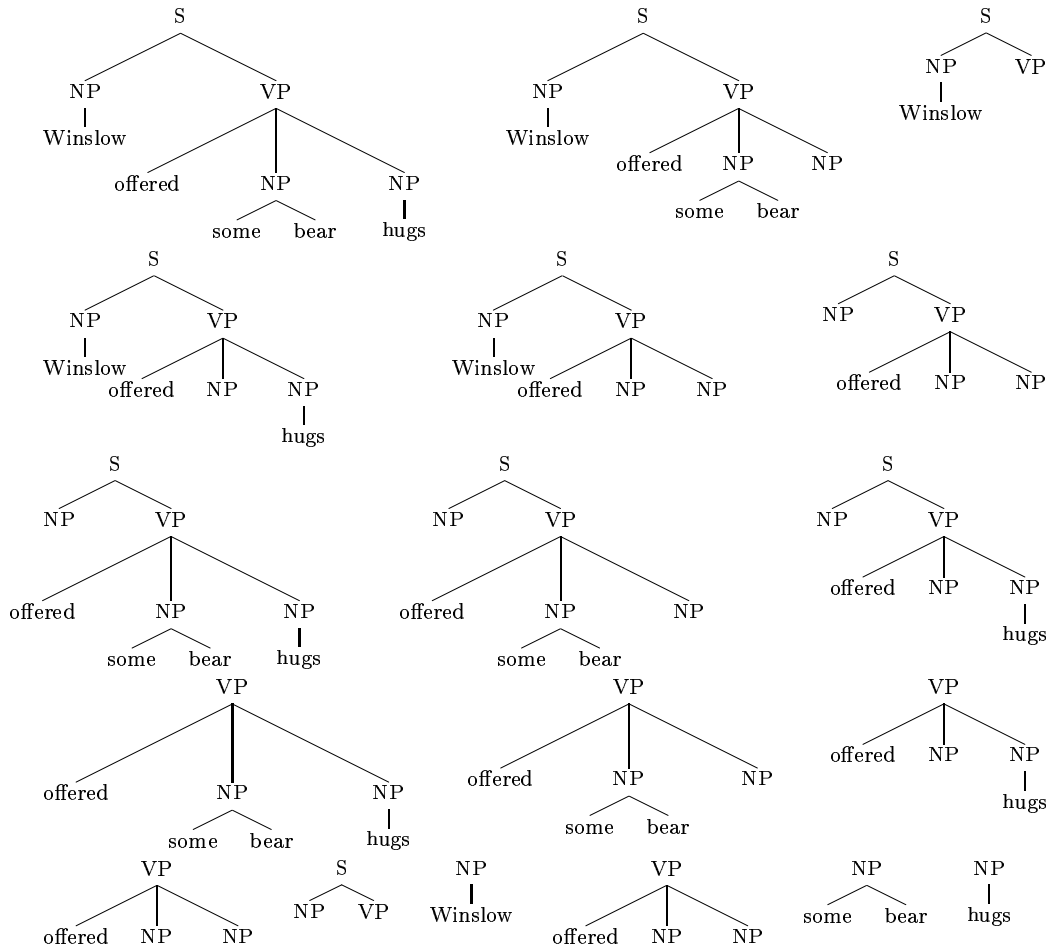


Figure 3.3: Treebank for Tree 3

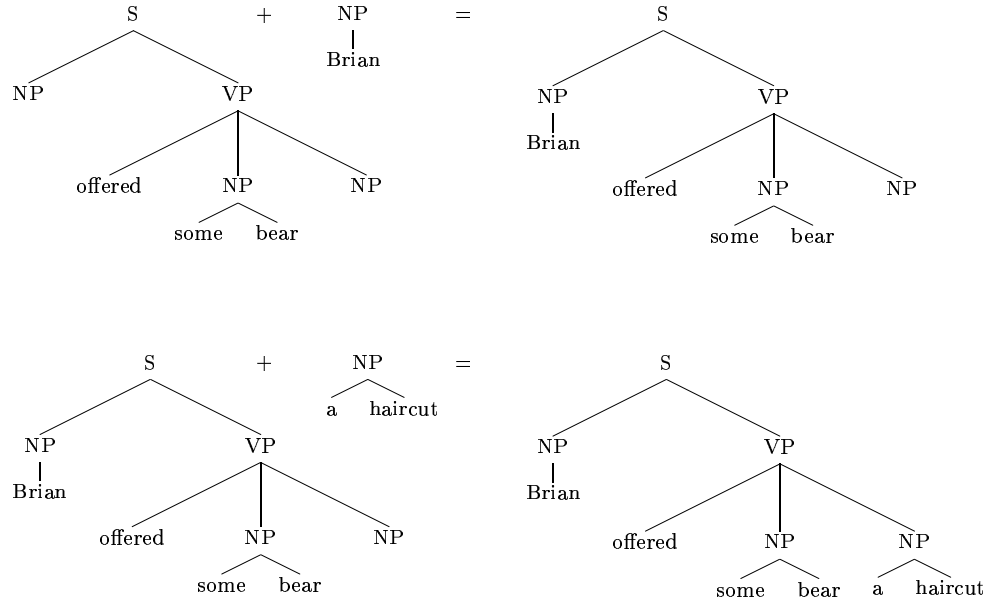


Figure 3.4: Derivation 1

Substructure	Occurrences	Occurrences of root-node	Relative Frequency
	1	20	0.05
	2	9	0.22

To compute the probability of a derivation, we multiply the probabilities (i.e. the relative frequencies) of its fragments. Figures 3.4 and 3.5 showed two possible derivations for the parse of the sentence *Brian offered some bear a haircut*, the probability of which are computed as follows:

$$\text{Derivation 1 (Fig 3.4)} \quad 1/20 * 1/7 * 1/7 = 0.0010$$

$$\text{Derivation 2 (Fig 3.5)} \quad 3/20 * 1/7 * 2/9 * 1/7 * 1/7 = 9.72 \times 10^{-5}$$

To compute the probability of the parse itself, we need to generate all possible derivations of that parse and compute their probabilities. The sum

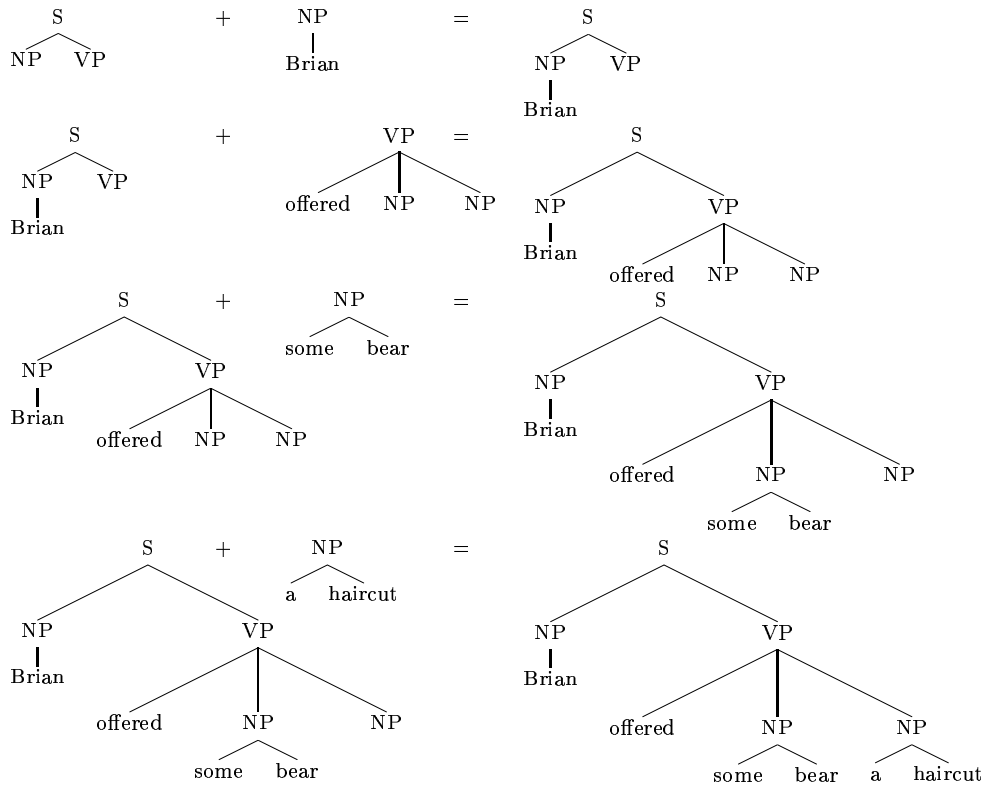


Figure 3.5: Derivation 2

of these constitutes the probability of the parse itself.

Typically however, a sentence has many different parses and it would be computationally hardly tractable to consider all derivations for each parse. VITERBI optimization⁴ can be considered to find the best parse using a best-first strategy. [Bod 1998] however shows that VITERBI only succeeds in finding the most probable derivation as opposed to the most probable parse, which limits parsing accuracy to 69% (as opposed to 85% (cf. Table 3.1)) in experiments on the ATIS-corpus. It is an important notion that the most probable derivation in the collection of derivation forests of the parse forest does not necessarily equal the most probable parse in the parse forest, as it causes many of the computational efficiency issues surrounding DOP.

3.1.2 Experimental Results of DOP

The basic DOP-model, DOP1, was tested on a manually edited version of the ATIS-corpus [Marcus et al. 1994]. The system was trained on 603 sentences (part-of-speech tag sequences) and evaluated on a test set of 75 sentences. Exact Match accuracy was used as an evaluation measure, expressing the percentage of sentences in the test set for which the parse proposed by the system is completely identical to the gold-standard parse, i.e. the tree-structure featured in the original corpus.

Different experiments were conducted in which maximum substructure size was varied. With DOP1-limited to a substructure-size of 1 (equivalent to a PCFG), exact match accuracy is 47%. In the optimal DOP-model, in which substructure-size is not limited, an exact match accuracy of **85%** is obtained. Table 3.1 displays exact match accuracy scores for the DOP-model for different experiments in which maximum substructure size was varied (scores reported in [Bod 1998]).

⁴Simply put: a best-first search through the search space.

Maximum depth of subtrees	Exact Match Accuracy	
	MONTE CARLO Optimization	Most probable Derivation
1	47%	47%
<2	68%	56%
<3	79%	65%
<4	83%	67%
<5	84%	67%
<6	84%	69%
unbound	85%	69%

Table 3.1: Experimental Results DOP1

3.2 Computational Efficiency of DOP

Looking for the most probable parse using the MONTE CARLO estimation technique, clearly yields better results than simply looking for the most probable derivation. Table 3.1 shows that there is no difference in exact match accuracy between the two techniques when substructure size is limited to 1, since DOP is then equivalent to a PCFG, in which there is only one possible derivation for any given parse. Whereas there is indeed a one-to-one mapping between derivation and parse in a PCFG-system, the task of finding the most probably parse does not equal to finding the most probable derivation in the DOP-framework⁵. As a consequence, all possible derivations of all possible parses for a particular sentence would need to be considered. This is not very efficient from a computational point-of-view, so that the MONTE CARLO algorithm needs to be introduced as an approximation to finding the most probable parse.

DOP1 in its optimal form achieves a very high exact match accuracy. The computational costs of the system, however, are equally high. [Bod 1995] reported an average parse time of 3.5 hours per sentence. Even though current parse times are more reasonable, the optimal DOP algorithm in which no constraints are made on the size of substructures to be considered, is

⁵[Sima'an 1996; Sima'an et al. 1994] proves that finding the most probably parse in the DOP-framework is an NP-complete problem, while finding the most probable derivation can be solved efficiently in a VITERBI-like manner [Sima'an 1999].

still computationally very expensive. And even though the most probable derivation can be found in polynomial time [Sima'an 1999], the problem of finding the most probable parse still constitutes a computational bottleneck. This is however not an issue for PCFGs, which makes them more efficient in that respect, albeit with a significantly lower degree of accuracy.

Although the use of larger syntactic contexts is highly relevant from a psycholinguistic point-of-view, it is by no means clear that language-users actually consider multiple derivations of the same parse when processing utterances. Furthermore, neither the original DOP-framework, nor the optimization proposed by [Sima'an 1999] seem to make an explicit reference to the internal processing of larger substructures, as is evident in human natural language processing. While the MONTE CARLO optimization scheme for instance maximizes the probability of the derivations and seems to prefer derivations made up of larger substructures, this effect occurs almost as a side-effect of the probabilistic processes involved. It may be interesting to see if we can make this preference for larger substructures more explicit by enhancing the memory-based aspects of the DOP-model.

Whereas most optimization methods for the DOP-model seem to focus on limiting the number of derivations to be considered or reducing the task of finding the most probable parse to the task of finding the most probable derivation as far as possible, one might consider the following syntactically inspired, rather than probabilistic optimization: by evaluating a single parse in terms of its similarity to large, previously observed constructions, we make the memory-based aspect, rather than the probabilistic operator the focal point of the processing stage. This chapter will show how it is possible to develop this method, without having to consider multiple derivations for a single parse, thereby simply equating the problem of finding the most probable parse to finding the most probable derivation. This allows us to exploit the attractive DOP-attribute of context-sensitivity and combine it with the computational efficiency of PCFGs.

3.3 Pattern-matching: A Memory-Based Approach

In this section, we take a look at a memory-based approach to syntactic analysis in the form of the Pattern-Matching Probabilistic Grammar (PMPG). This re-interpretation of the DOP model amplifies the *nearest neighbor* and similarity aspects of MBL by analyzing a parse in terms of its similarity to syntactic *patterns* stored in memory to which it can be matched (cf. infra). *Pattern-Matching*, a basic concept in statistical as well as symbolical approaches to machine learning, is used in the context of this chapter to refer to the method of using syntactic patterns stored in memory to evaluate a parse.

3.3.1 Memory-Based Parsing

The Memory Based aspects of DOP have already been made more explicit in [Scha 1999] and [Argamon et al. 1999], but we would like to concentrate on the following properties that DOP shares with MBL:

Trivial Knowledge Acquisition Phase

Knowledge acquisition in the DOP-framework only consists of storing fragments of syntactic structures in memory. Even though the fragments have a probabilistic weight attached to them, no further supra-segmental information is extrapolated in the form of decision tree structures and the like.

Emphasis on Memory

Whereas other parsing methods concentrate on optimizing the performance of PS-grammars, DOP aspires to be an implementation of memory-based processing, which tries to corroborate the psycholinguistically motivated claim that language users employ previous language experience stored in memory, while processing language [Chandler 1992; Skousen 1989]⁶.

Robustness with respect to low-frequency events

Since all syntactic substructures are stored in memory, even those who seem to constitute noise, DOP is a robust parsing method that is able to generate marginal, low-frequency structures.

Whereas the current DOP-models indeed have a MBL-type acquisition phase, the classification task is quite dissimilar: when classifying new data, there is only indirect reference to the stored examples in the probabilistic disambiguation of the derivation forest. As we have mentioned before, parsing with the DOP-model involves randomly generating derivations and looking at the most common tree in the forest. This is different from a MBL-approach in that there is no real look-up of the nearest neighbor in memory. DOP does not traverse memory in search of the most similar item in memory, so that it can extrapolate its solution.

Even though it is not possible for full syntactic parsing to employ such a direct look-up procedure in the way a feature value-based classification task can, a more direct implementation of the memory-based aspects of DOP is possible. When we look at natural language parsing from a memory-based point of view, one might say that a sentence can be analyzed by looking for the most similar sentence in memory and by consequently extrapolating its tree-structure.

The parsing system described in this chapter tries to mimic this behavior

⁶Note that the psycholinguistic argument does provide a paradox: if language users employ previously registered *constructions* to analyze sentences, how did they come to store these in the first place?

by explicitly interpreting the DOP-model as a memory-based model, in which analyses are being matched with syntactic *patterns* recorded in memory. This is achieved by evaluating each parse in the parse forest in terms of its similarity to the structures in memory. This poses two problems: (i) what is the level on which we should perform our nearest neighbor look-up operation and (ii) how do we compute similarity between grammatical structures?

The level of matching

A rigid MBL-approach would involve finding a sentence in memory that is similar to the sentence to be parsed and consequently extrapolating its tree-structure. Similar sentences such as *I want a drink* and *I want to drink* have totally different grammatical structures however and simply employing the tree-structure of the highly similar sentence would yield bad parsing accuracy. Furthermore, although this point is moot [Chomsky 1957], there is a coverage problem in that we would need an unlikely huge corpus to account for all possible productions of sentences and tree-structures.

The nearest neighbor approach employed in classification tasks that can be described using data in a propositional format, cannot be used for syntactic parsing. A parsing task involves two tasks, i.e. (a) finding the constituent boundaries and (b) finding the correct category for the clauses. As such, parsing can only be achieved by either cascading classifiers using propositional data [Daelemans et al. 1999; S. Buchholz 1999; Tjong Kim Sang 2002], or by introducing a method to deal with new unseen productions, so that the aforementioned coverage issue can be resolved.

By bringing down the lookup-operation to a lower level, i.e. the constituent level, we do not only make our parser more robust in terms of coverage, but we also create an opportunity to produce new syntactic structures that were previously unavailable in memory. The importance of this feature cannot be stressed enough, since the goal of our parsing system is indeed to parse new, unseen sentences. It is therefore inadvisable to employ a rigid application of the memory-based lookup procedure alone, without implementing a possibility for extra, subsegmental lookup procedures.

Similarity

Now that we have established the structural level on which we are basing our comparison, we need a way to compute similarity between structures. We will define the similarity of a parse in terms of the *patterns* of which it is composed. Finding the nearest neighbor of a given parse will be done by *pattern-matching*, the exact procedure of which will be outlined in Section 3.3.2.

For each given parse in the parse forest⁷, its similarity to the structure in memory can be expressed by computing the number of nodes that can be retrieved from memory. Note that a one-to-one comparison is not being made between a particular parse and the analyses recorded in memory, as is the case for memory-based learners, such as TIMBL. As the basis of our comparison is on the level of the clause, we allow the memory-based learner to construct its own analysis from patterns from tree-structures in memory. The memory-based component will try to find the minimal combination of the largest chunks of syntactic data to construct the most similar structure to the proposed parse.

Similarity between the proposed analysis and the patterns in memory is computed by a combination of the following features:

- the number of patterns needed to construct a tree (to be minimized)
- the size of the patterns that are used to construct a tree (to be maximized)

The *nearest neighbor* for a given analysis can therefore be defined as the derivation that contains the largest substructures that can be retrieved from memory and therefore shares the most nodes with previously observed analyses. Sections 3.7 and 3.8 will more formally define how the number of patterns and the size of the patterns are combined into one probabilistic similarity measure.

Now that we have established the heuristics for our memory-based parsing

⁷The experiments outlined in this chapter used the CYK-parser described in [Chappelier and Rajman 1998] to generate parse forests.

model, we will take a closer look at the formal encoding of the pattern-matching procedure.

3.3.2 Compiling Contextual Information

Before we can actually use syntactic substructures as patterns for our look-up procedure, we first need to introduce a way of encoding grammatical contextual information.

Training

To compile parse trees into patterns, all substructures in the training set are encoded by assigning them specific indices. This approach was inspired by [Goodman 1996], in which a system of indexed parse trees is (unsuccessfully) used to reproduce DOP-like performance as an equivalent PCFG. The system of indexing used in our experiments (which is described in more detail in [De Pauw 2000b]), is however specifically geared toward encoding any type of contextual information in parse trees.

During the training phase, tree-structures from the training set are indexed in a bottom-up fashion. Let us consider the parse tree in Figure 3.6 for this typical ATIS-corpus sentence: “*XXX Show me the flights from Washington DC to San Francisco early Wednesday*” and the ps-grammar that was used to construct this parse in terms of levels, by horizontally slicing the parse tree⁸:

⁸Check Appendix H for the meaning of category labels and part-of-speech tags.

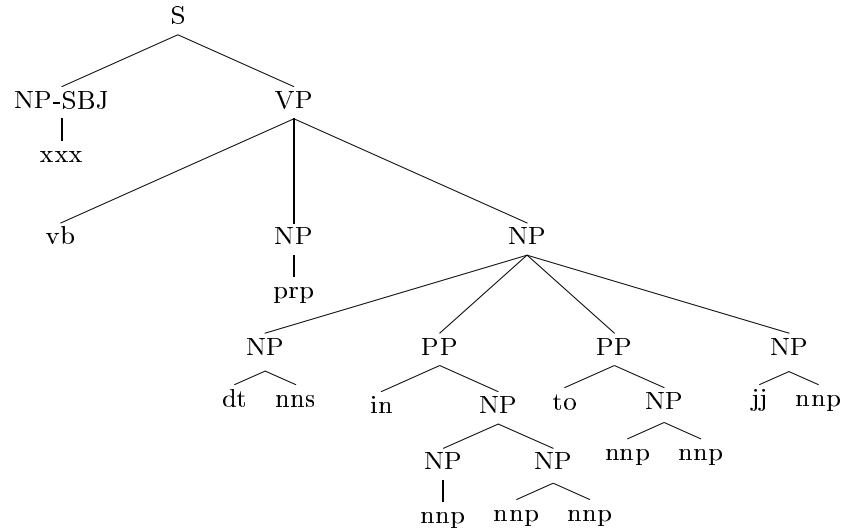


Figure 3.6: Parse-tree for the sentence *XXX Show me the flights from Washington DC to San Francisco early Wednesday*

Level 1	S	→	NP-SBJ VP
Level 2	NP-SBJ	→	xxx
	VP	→	vb NP NP
Level 3	NP	→	prp
	NP	→	NP PP PP NP
Level 4	NP	→	dt nns
	PP	→	in NP
	PP	→	to NP
	NP	→	jj nnp
Level 5	NP	→	NP NP
	NP	→	nnp nnp
Level 6	NP	→	nnp
	NP	→	nnp nnp

Next, we iteratively traverse the nodes in a bottom-up fashion, observing the contents of the nodes. If a node is found that contains only terminals, it is given an index. This index is either retrieved from memory if that particular pattern has already been observed (and previously indexed) in the data, or a new index is created and stored in memory. Starting at the bottom-level 6, we notice that both nodes at Level 6 consist of terminals. We have not

previously seen this type of node, so we provide it with a new index:

Level 6 NP@2 → nnp
 NP@1 → nnp nnp

The indexed categories and the context which they represent are stored in memory. This comes in handy when indexing the nodes on level 5. The first node $NP \rightarrow NP NP$ contains non-terminals and is therefore not ready for indexing. The second node however does contain only terminals. Furthermore, this syntactic structure has already been registered before, enabling us to use the previously registered index for this node:

Level 5 NP → NP NP
 NP@1 → nnp nnp

This process is repeated for each level in a bottom-up fashion, yielding the following updated grammar:

Level 1 S → NP-SBJ VP
Level 2 NP-SBJ@6 → xxx
 VP → vb NP NP
Level 3 NP@5 → prp
 NP → NP PP PP NP
Level 4 NP@4 → dt nns
 PP → in NP
 PP → to NP
 NP@3 → jj nnp
Level 5 NP → NP NP
 NP@1 → nnp nnp
Level 6 NP@2 → nnp
 NP@1 → nnp nnp

We now consider the indexed nodes as terminals, by pruning the parse tree (Figure 3.7) and percolating the indexed node to the higher levels in the grammar:

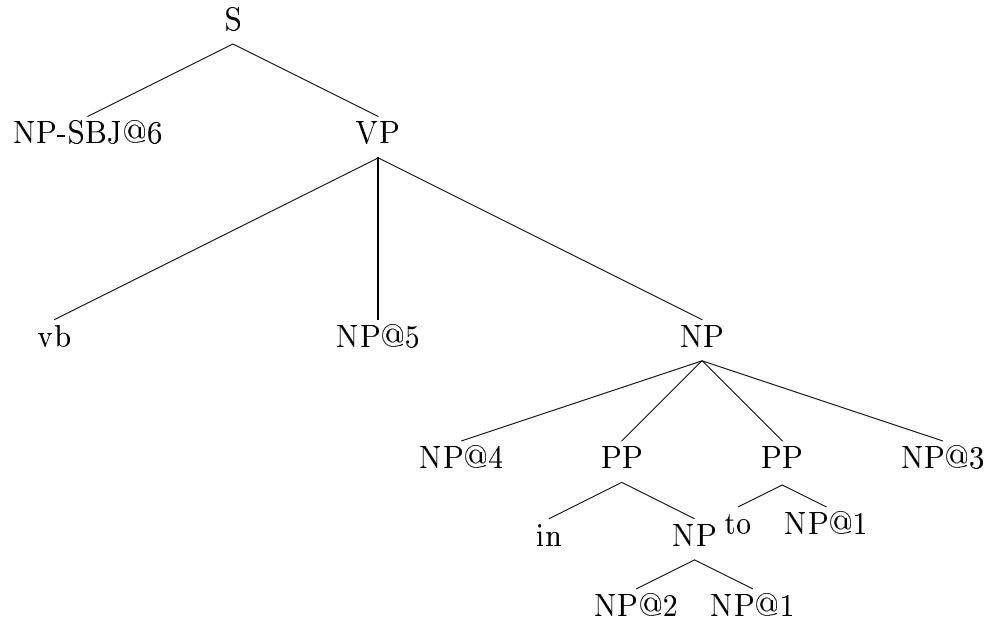


Figure 3.7: Pruned Parse-Tree

Level 1	S	→	NP-SBJ@6 VP
Level 2	NP-SBJ@6	→	xxx
	VP	→	vb NP@5 NP
Level 3	NP@5	→	prp
	NP	→	NP@3 PP PP NP@4
Level 4	NP@4	→	dt nns
	PP	→	in NP
	PP	→	to NP@1
	NP@3	→	jj nnp
Level 5	NP	→	NP@2 NP@1
	NP@1	→	nnp nnp
Level 6	NP@2	→	nnp
	NP@1	→	nnp nnp

Next, the indexing process resumes. The node $NP \rightarrow NP@2 NP@1$ at Level 5 will also be indexed, since it contains two indexed nodes, which are considered to be terminals. After indexing and percolation of indexed nodes, we obtain the following grammar and the pruned tree in Figure 3.8.

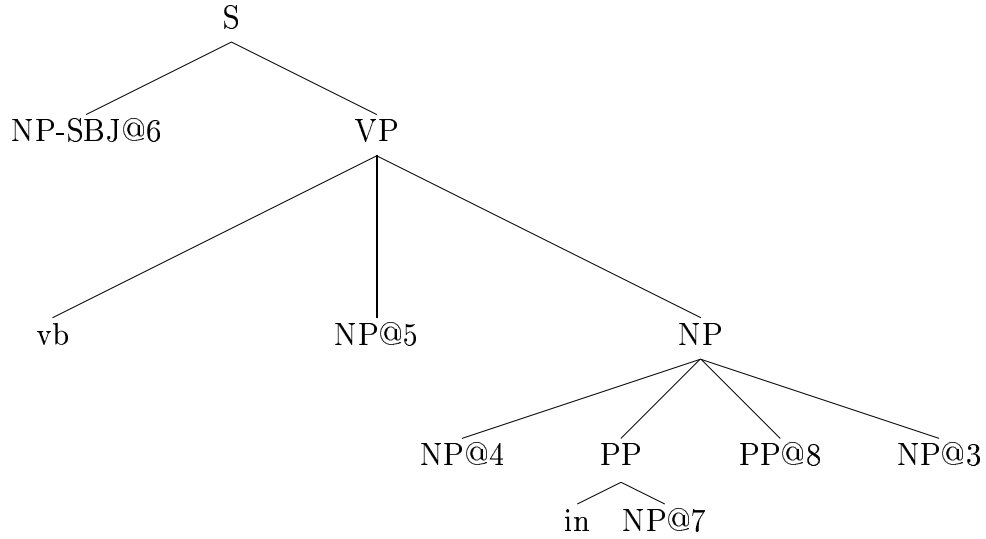


Figure 3.8: Pruned Parse-Tree 2

Level 1	S	→	NP-SBJ@6 VP
Level 2	NP-SBJ@6	→	xxx
	VP	→	vb NP@5 NP
Level 3	NP@5	→	prp
	NP	→	NP@4 PP PP@8 NP@3
Level 4	NP@4	→	dt nns
	PP	→	in NP@7
	PP@8	→	to NP@1
	NP@3	→	jj nnp
Level 5	NP@7	→	NP@2 NP@1
	NP@1	→	nnp nnp
Level 6	NP@2	→	nnp
	NP@1	→	nnp nnp

The process of indexing and percolation iterates until the top-node is indexed. This yields the following grammar:

Level 1	S@12	→	NP-SBJ@6 VP@11
Level 2	NP-SBJ@6	→	xxx
	VP@11	→	vb NP@5 NP@10
Level 3	NP@5	→	prp
	NP@10	→	NP@4 PP@9 PP@8 NP@3
Level 4	NP@4	→	dt nns
	PP@9	→	in NP@7
	PP@8	→	to NP@1
	NP@3	→	jj nnp
Level 5	NP@7	→	NP@2 NP@1
	NP@1	→	nnp nnp
Level 6	NP@2	→	nnp
	NP@1	→	nnp nnp

A single indexed category holds a lot of contextual information. In this grammar for instance, NP@10 indicates the entire NP-structure for the constituent *the flights from Washington DC to San Fransisco early Wednesday*. The original grammar can now be extended to encode this contextual information, as is shown in Table 3.2. This grammar can be fed to any parsing method able to deal with context-free grammars (see Section 3.11).

Note that this encoding scheme differs from the one outlined in [Goodman 1996]) in two respects: first, it does not share the binary branching method [Goodman 1996] employs. Second, in this scheme contextual information is encoded in a bottom-up fashion. VP@11 only gives us lower-bound contextual information. It does not provide information about the upper syntactic context. This makes our encoding scheme more flexible and makes bigger substructures more available for the construction of new, unseen tree-structures.

Testing

Indexing during the testing phase only differs from the training phase in that no new indices are created when confronted with new and unseen substructures. Consider the next example: in the parse forest for the sentence *the screens display the flights from Washington D C to Los Angeles next Friday*, we find the (correct) parse in Figure 3.9. This tree-structures seems to be almost identical to the one in Figure 3.6, except for the NP-SBJ-node.

Original	PMPG	
S → NP-SBJ VP	S@12 → NP-SBJ@6 VP@11	S → NP-SBJ@6 VP@11
	S@12 → NP-SBJ VP@11	S → NP-SBJ VP@11
	S@12 → NP-SBJ@6 VP	S → NP-SBJ@6 VP
	S@12 → NP-SBJ VP	S → NP-SBJ VP
NP-SBJ → prp	NP-SBJ@6 → prp	NP-SBJ → prp
VP → vbp NP	VP@11 → vb NP@5 NP@10	VP → vb NP@5 NP@10
	VP@11 → vb NP NP@10	VP → vb NP NP@10
	VP@11 → vb NP@5 NP	VP → vb NP@5 NP
	VP@11 → vb NP NP	VP → vb NP NP
NP → prp	NP@5 → prp	NP → prp
NP → NP PP PP NP	NP@10 → NP@4 PP@9 PP@8 NP@3	NP → NP@4 PP@9 PP@8 NP@3
	NP@10 → NP@4 PP@9 PP@8 NP	NP → NP@4 PP@9 PP@8 NP
	NP@10 → NP@4 PP@9 PP NP@3	NP → NP@4 PP@9 PP NP@3
	NP@10 → NP@4 PP@9 PP NP	NP → NP@4 PP@9 PP NP
	NP@10 → NP@4 PP PP@8 NP@3	NP → NP@4 PP PP@8 NP@3
	NP@10 → NP@4 PP PP@8 NP	NP → NP@4 PP PP@8 NP
	NP@10 → NP@4 PP PP NP@3	NP → NP@4 PP PP NP@3
	NP@10 → NP@4 PP PP NP	NP → NP@4 PP PP NP
	NP@10 → NP PP@9 PP@8 NP@3	NP → NP PP@9 PP@8 NP@3
	NP@10 → NP PP@9 PP@8 NP	NP → NP PP@9 PP@8 NP
	NP@10 → NP PP@9 PP NP@3	NP → NP PP@9 PP NP@3
	NP@10 → NP PP@9 PP NP	NP → NP PP@9 PP NP
	NP@10 → NP PP PP@8 NP@3	NP → NP PP PP@8 NP@3
	NP@10 → NP PP PP@8 NP	NP → NP PP PP@8 NP
	NP@10 → NP PP PP NP@3	NP → NP PP PP NP@3
	NP@10 → NP PP PP NP	NP → NP PP PP NP
NP → dt nns	NP@4 → dt nns	NP → dt nns
PP → in NP	PP@9 → in NP@7	PP → in NP@7
	PP@9 → in NP	PP → in NP
PP → to NP	PP@8 → to NP@2	PP → to NP@2
	PP@8 → to NP	PP → to NP
NP → jj nnp	NP@3 → jj nnp	NP → jj nnp
NP → NP NP	NP@7 → NP@2 NP@1	NP → NP@2 NP@1
	NP@7 → NP@2 NP@1	NP → NP@2 NP@1
	NP@7 → NP@2 NP	NP → NP@2 NP
	NP@7 → NP NP@1	NP → NP NP@1
	NP@7 → NP NP	NP → NP NP
NP → nnp nnp	NP@1 → nnp nnp	NP → nnp nnp
NP → nnp	NP@2 → nnp	NP → nnp
NP → nnp nnp	NP@1 → nnp nnp	NP → nnp nnp

Table 3.2: Extended Grammar

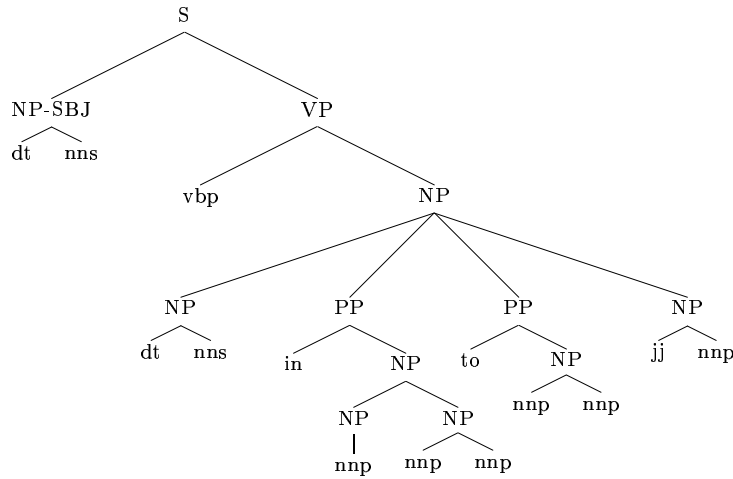


Figure 3.9: Parse-tree for the sentence *the screens display the flights from Washington DC to Los Angeles next Friday*

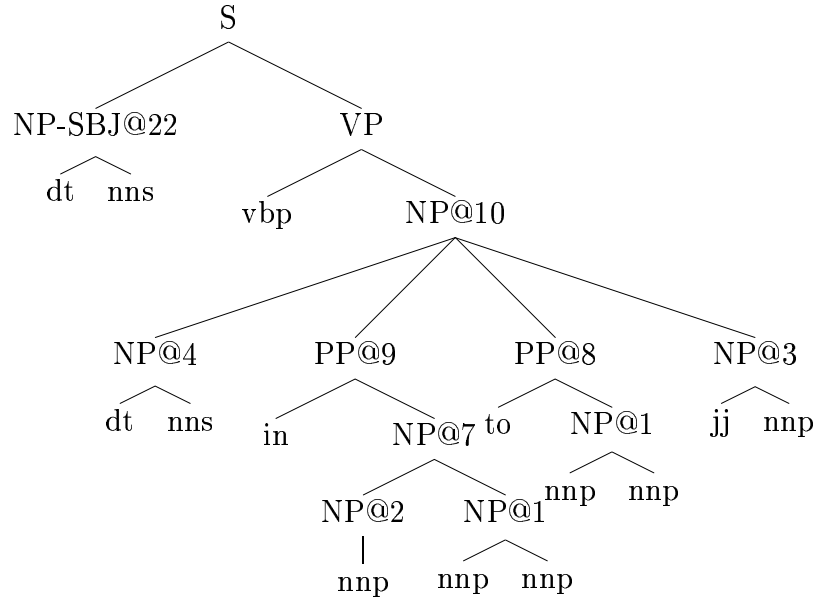


Figure 3.10: Indexed Parse-tree

Similarly to the training process, we index this tree-structure in a bottom-up fashion, this time however without producing new indices for unseen structures. When a substructure can be retrieved from memory, we index the top-node of that substructure and prune everything below the node. Assuming that somewhere in our training material we have observed a node $\text{NP-SBJ} \rightarrow \text{DT NNS}$ which we had indexed as NP-SBJ@22 , we can index the parse-tree as in Figure 3.10.

In this example we see that the object of the sentence, a fully specified NP, has been completely retrieved from memory, meaning that the NP in

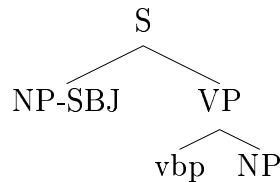


Figure 3.11: Pruned Parse-tree

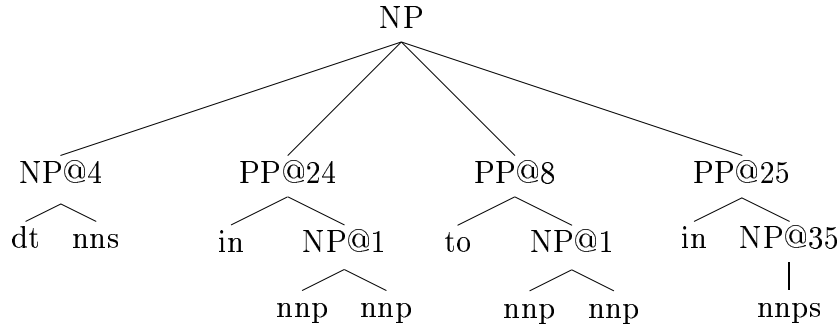


Figure 3.12: NP-structure

its entirety has been observed in the training set. However, no VP-node was found in memory that consists of that particular NP as an object to a VBP. In other words, there is no grammar rule $VP \rightarrow VBP\ NP@10$ in the extended grammar (cf. Table 3.2). The VP can therefore not be indexed and consequently the S-node as well.

Disambiguating with PMPG consequently involves pruning all substructures that can be retrieved from memory. This results in the parse-tree in Figure 3.11. Finally, the probability for this pruned parse tree is computed in a PCFG-type manner (not adding the retrieved nodes to the product):

$$P(\text{parse}) = P(S \rightarrow NP\text{-}SBJ\ VP) \cdot P(VP \rightarrow vbp\ NP)$$

An extension: partly matched structures

An important variation to the PMPG algorithm makes it more intuitive with respect to syntactic pattern-matching. Consider the example in Figure 3.12, which represents the syntactic-structure for *the flights from Los Angeles to San Francisco on Wednesdays*.

The indexing procedure has found the subordinate substructures in memory. But it has not found an NP-structure consisting of those particular substructures. After pruning the indexed structure, we are left with the following rule:

$NP \rightarrow NP@4 PP@24 PP@8 PP@25$

In the PMPG-algorithm described so far we would simply calculate the probability of the following rule:

$NP \rightarrow NP PP PP PP$

But in this kind of calculation, no distinction is being made between partly matched clauses and clauses that have no matched subordinates. Figure 3.13 exemplifies the problem. From a probabilistic point of view, structure (a) is just as likely as structure (b) and there is no indication of the fact that some subordinate clauses in structure (a) can be matched in this kind of structure (indicated by boxed nodes).

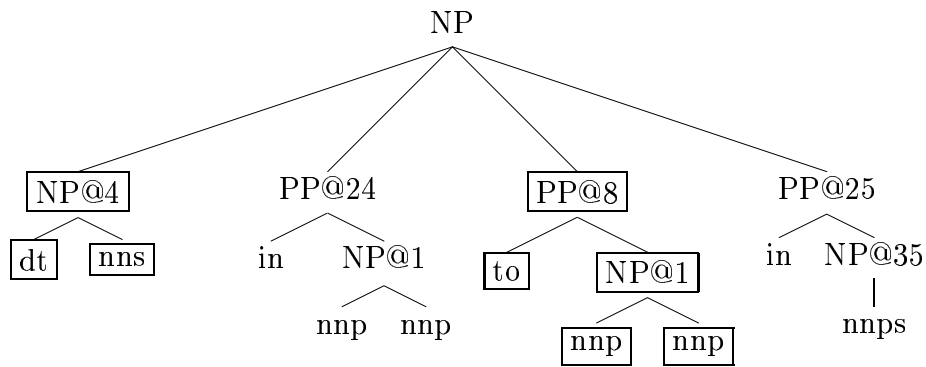
So we need to introduce a way to prefer rule $NP \rightarrow NP@4 PP PP@8 PP$ over $NP \rightarrow NP PP PP PP$. However, since the probability of the first rule is necessarily equal or lower than the probability of rule 2, we cannot simply use the probabilities attached to these rules.

We therefore need to tweak the calculation of partly matched structures, so that they will be preferred. Let us resume our example:

$NP \rightarrow NP@4 PP@24 PP@8 PP@25$

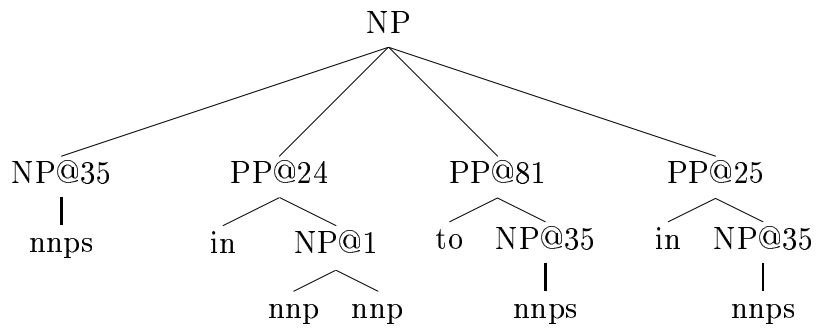
The indexing in this rule indicates that an NP-structure has been found of which all the subordinate clauses can be retrieved from memory, but that there was no NP-structure stored in memory containing these particular clauses. We then list all the permutations of this indexed rule:

(a) NP-structure with partly matched subordinate clauses



= NP → NP@4 PP PP@8 PP

(b) NP-structure with no matched subordinate clauses



= NP → NP PP PP PP

Figure 3.13: 2 different NP-structures

NP \rightarrow NP@4 PP@24 PP@8 PP@25
 NP \rightarrow NP@4 PP@24 PP@8 PP
 NP \rightarrow NP@4 PP@24 PP PP@25
 NP \rightarrow NP@4 PP@24 PP PP
 NP \rightarrow NP@4 PP PP@8 PP@25
 NP \rightarrow NP@4 PP PP@8 PP
 NP \rightarrow NP@4 PP PP PP@25
 NP \rightarrow NP@4 PP PP PP
 NP \rightarrow NP PP@24 PP@8 PP
 NP \rightarrow NP PP@24 PP@8 PP
 NP \rightarrow NP PP@24 PP PP@25
 NP \rightarrow NP PP@24 PP PP
 NP \rightarrow NP PP PP@8 PP@25
 NP \rightarrow NP PP PP@8 PP
 NP \rightarrow NP PP PP PP@25
 NP \rightarrow NP PP PP PP

Next we look up these rules in the grammar, such as the one in Table 3.2, and sum the probabilities of the rules that can be retrieved. This way, we take the base probability of the rule $\text{NP} \rightarrow \text{NP PP PP PP}$ and add the probability of any partly matched structures, thereby expressing a preference for (partly) matched structures.

To summarize, PMPG can be defined as having two basic operators. The Pattern-Matching (i.e. Memory Based) part of PMPG greedily looks for similar structures in memory. For each node in the parse, PMPG looks for an identical node in memory. Inherently, the similarity of a parse and its (compound) nearest neighbor is calculated by counting the number of nodes they share. In this instantiation, PMPG assigns a probabilistic weight of 1.0 to a retrieved node, thereby, in actuality, pruning it from the parse tree. The probabilistic operator of PMPG consequently computes the probability of the remainder of the parse tree. In the infrequent case of a tie between parses, PMPG randomly picks a candidate.

3.4 Experimental Setup

We now look at a set of experiments to evaluate the PMPG method against the standard PCFG-algorithm. In this section, we will describe the experimental setup.

The experiments were conducted on two different corpora of the Penn Treebank [Marcus et al. 1993]: (i) the ATIS-corpus and (ii) the Wall Street Journal corpus (WSJ). 10-fold cross-validation (10xv) was used to test the parsers: the tree-structures of the annotated corpus were randomly divided over 10 partitions of equal size. Each partition was used as a test partition, with the other nine partitions being used as training partitions. This approach leads to more reliable results, as it reduces the possibility of the results being influenced by an (un)favorable test/training set partitioning.

The main experiments were carried out on an edited version of the ATIS-II-corpus, which consists of 578 sentences. The ATIS-corpus is a small, relatively homogeneous set of sentences used by people perusing the Air Travel Information System and therefore constitutes a corpus of transcribed and annotated spoken utterances [Hemphill et al. 1990]. Quite a lot of annotation errors and inconsistencies were found in the ATIS corpus, but not corrected, since we want our (probabilistic) system to be able to deal with this kind of noise. Semantically oriented flags like -TMP and -DIR, most often used in conjunction with PP, were removed, since there is no way of retrieving this kind of semantic information from the part-of-speech tags of the ATIS-corpus. Syntactic flags like -SBJ, on the other hand, have been maintained. Internal relations (denoted by numeric flags) were removed. There was no limit on sentence length.

The experiments on the Wall Street Journal-corpus were limited to (i) a 10xv setup on section 1 and (ii) a setup consisting of the frequently used division that takes sections 2 to 21 as the training set and section 23 as the test set [Collins 1997; Charniak 1997; Ratnaparkhi 1999; Bod 2000]. The same orthographic adjustments were made on the Wall Street Journal Corpus as on the ATIS-corpus. This makes it harder to compare the experiments of the WSJ-corpus with other work in the field [Collins 2000a; Collins 1997; Bod 2000; Ratnaparkhi 1997], which typically strip all semantic flags, co-reference information and quotation marks and merge the ADVP and PRT categories.

We believe the orthographic adjustments made on our data set to be more equitable with regards to the parsing task, even though it negatively affects accuracy scores.

Figure 3.14 schematically displays the experimental setup. An annotated treebank is divided in a training set and a test set (usually at a 90% to 10% ratio). A context-free grammar is induced from the training set, which powers a CYK parser [Chappelier and Rajman 1998]. This parser produces parse-forests for the part-of-speech tag sequences of the test set. The tree-structures in these parse forests are not ranked and no preference for any particular parse is evident as yet.

Next, three disambiguating algorithms (henceforth the *disambiguators*), which also employ information culled from the training set, provide a ranking for the parses in the parse-forests. The first parse of this ordered parse forest is consequently the one that the disambiguator proposes to be the correct one.

3 disambiguators were tested:

- PCFG: a simple Probabilistic Context-Free Grammar (Section 3.6)
- PMPG: the DOP approximation, Pattern-Matching Probabilistic Grammar (Sections 3.3 and 3.7)
- PCFG+PMPG: a combined system, integrating PCFG and PMPG (Section 3.11)

The main evaluation measure we consider is exact match accuracy, but also the PARSEVAL measures, Labeled Precision (LR), Labeled Recall (LP) and the unweighted balance of the 2, $F_{\beta=1}$ -score will be provided. [van Rijsbergen 1975] defines the computation of these measures as follows:

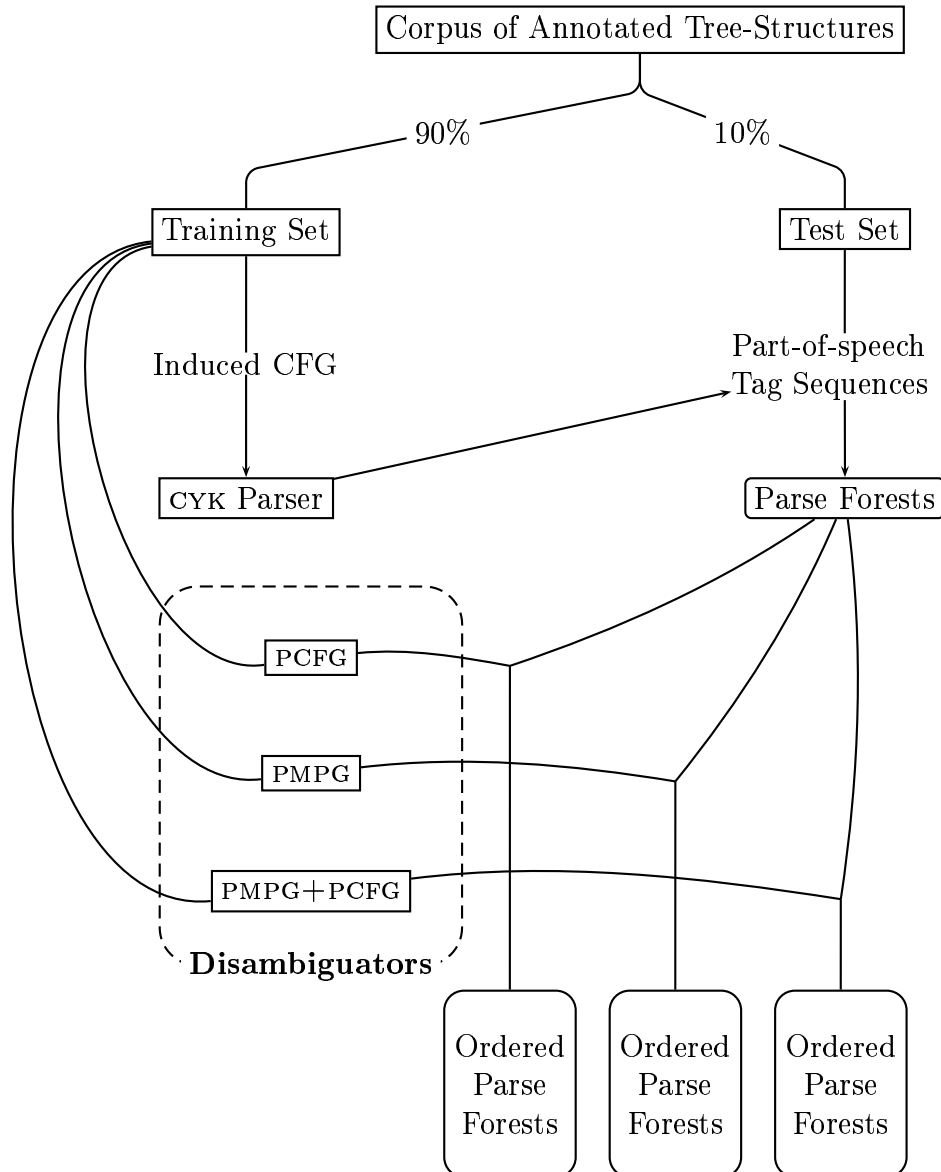


Figure 3.14: Experimental setup

C = number of correct constituents in parse
 P = number of constituents in Parse
 T = number of constituents in gold standard structure, i.e. original structure in annotated corpus

$$LP = \frac{c}{P} \qquad LR = \frac{c}{T}$$

$$F_{\beta=1} = \frac{(\beta^2+1)*LP*LR}{\beta^2*LR+LP} = \frac{2*LP*LR}{LP+LR}$$

3.5 The Parsing Phase

In the first processing stage, the parser analyzed each sentence in the test set. This initial stage proved to be quite problematic for the ATIS corpus, since overall, 35 out of the 578 sentences (6%) could not be parsed, because the grammar induced from the training set did not contain the proper grammatical information to trigger the correct analysis for a sentence in the test set. This constitutes a *sparse grammar* problem, which means that sparse data causes the grammar that is induced to be incomplete with respect to its coverage on the structures in the test set.

Note that the 6% figure does not even include those sentences for which some (erroneous) parse was found, but for which the correct parse could a priori not be found, since the required rewrite rule needed to construct the correct parse tree for a sentence in the test set, was not featured in the induced grammar. We identified 97 tree-structures (16.8%) in the ATIS-corpus that featured a rule that was unique to that structure, i.e. 97 sentences that can never be parsed correctly by an independent training set.

NP-annotation in particular seemed to be the main cause for unparsability. An NP like *restriction code AP/57* is represented by the rewrite rule:

NP \rightarrow NN NN sym sym sym CD CD

Highly specific and flat structures like these are very specific constructs for one particular sentence and are usually not featured in the grammar induced from an independent training set.

The most common method to deal with these flat structures is by employ-

ing Markov Grammars [Collins 1997] that calculate the probability of these structures by determining the head daughter-node of the constituent and its probability, after which the probabilities of its modifiers can be deduced as well.

Part II will deal with evolutionary approaches to grammatical smoothing as a possible solution to this problem. Chapter 7 in particular will describe a method that randomly creates new grammatical information, while evolutionary computing methods distinguish useful from erroneous grammatical information, thereby increasing grammar coverage without the need for external information sources.

One might also consider generating parse forests with an independent grammar, induced from the entire corpus (training set+test set). This option violates the blind-testing constraint by using grammatical information induced from the test set, but may appear justifiable if we are only interested in probabilistic ordering of parse forests and only use probabilistic information induced from the training set. By employing probabilistic smoothing this method can attribute a (usually low) probability to unseen rules, which in this case are rules induced from the test set. But the number of unknown rules in this case is not arbitrary, as it is relative to the number of known rules. The distribution of the probability mass reserved for the test set is therefore also directly related to the number of rules induced from the test set. The probabilistic smoothing method, which should have provided a way to perform probabilistic re-ordering of parse forest without exploiting knowledge from the test set, therefore also violates the blind-testing constraint.

Another way of generating parse forests with an independent grammar would be to induce one from a different, but similarly labeled corpus, e.g. the WSJ-corpus. This usually entails using grammatical information extracted from a corpus from a different domain, which would have an unfavorable effect on parsing accuracy. A way to deal with this would be, as in the previous method, to extract the grammatical information from the alternate corpus and use the training set to provide probabilistic weights for the rules. Again, a portion of the probability mass would need to be reserved to distribute among the other rules in the grammar. This method might yield better parsing accuracy, but it is nevertheless unlikely that specific domain-dependent structures such as $NP \rightarrow NN\ NN\ sym\ sym\ sym\ CD\ CD$ would be covered by the alternate grammar, leading us back to square one.

Neither grammatical, nor probabilistic smoothing was implemented in the context of the experiments described in this paper, but one should keep in mind that the accuracy of the parsing systems can be improved, if a more adequate grammar is supplied.

In any case, the sparseness of the grammar proves to be a serious bottleneck for exact match accuracy, limiting our disambiguators to an upper bound exact match accuracy of about 83.2%.

Parsing the more elaborate Wall Street Journal corpus is less problematic: the 10-fold cross validation on Section 1 only counted 15 unparseable sentences, which accounts for less than 1% of the sentences. Note again that this does not mean that for the remaining 99% of the sentences a grammatical parse was generated in the parse forest. [Collins 1999] reports that when using section 2-21 as a training set and section 23 test set, 17.1% of the sentences in the test set have a rule not seen in the training set, indicating that the grammar sparseness problem is still very much apparent, even though it is seemingly not expressed in the ratio of unparseable sentences.

3.6 PCFG-experiments

A PCFG computes the probability of a parse tree by multiplying the probabilities of the rewrite-rules that were used to construct the parse. Note that a PCFG is identical to DOP1 when we limit the maximum depth of the substructures size to 1. Any given parse consequently has one and only one possible derivation, so that finding the most probable derivation entails finding the most probable parse as well (cf. Section 3).

The full results of the experiments can be consulted in Appendix B. Table 3.3 provides an overview of the results on the ATIS-corpus. The first line of Table 3.3 shows the results for the PCFG-experiment: **60%** exact match accuracy for the ATIS-corpus is an adequate result for this baseline model. Precision is respectable at 88.0%, but recall is trailing at 81.9% ($F_{\beta=1}$ score of **84.8%**). During experimentation, we noticed that two partitions scored significantly lower on exact match accuracy and precision and recall measures (cf. Appendix B). This was due to an unfavorable data partition, which causes the aforementioned grammar sparseness to come into play, as well as

ATIS - 10 Fold Cross Validation					
Parser	Exact Match (%)	LP	LR	$F_{\beta=1}$ (%)	sec
PCFG	60.0 (± 8.3)	88.0 (± 3.2)	81.9 (± 5.4)	84.8 (± 4.7)	0.04
PMPG	68.3 (± 6.1)	85.6 (± 4.1)	85.3 (± 4.1)	85.4 (± 3.9)	0.16
PCFG+PMPG	74.0 (± 7.6)	92.7 (± 1.9)	88.6 (± 3.7)	90.6 (± 5.1)	1.00

Table 3.3: Experimental Results ATIS - Overview

Wall Street Journal - 10 Fold Cross Validation (Section 1)					
Parser	Exact Match (%)	LP	LR	$F_{\beta=1}$ (%)	sec
PCFG	8.5 (± 2.1)	64.8 (± 4.6)	64.3 (± 1.4)	64.5 (± 2.7)	3.2
PMPG	12.3 (± 2.0)	67.0 (± 1.9)	66.4 (± 1.3)	66.7 (± 1.4)	34
PCFG+PMPG	14.9 (± 2.2)	83.4 (± 3.2)	80.5 (± 2.1)	81.9 (± 2.6)	98

Table 3.4: Experimental Results WSJ 10xv - Overview

provide a grammar whose probabilistic distribution is less suited for the test set.

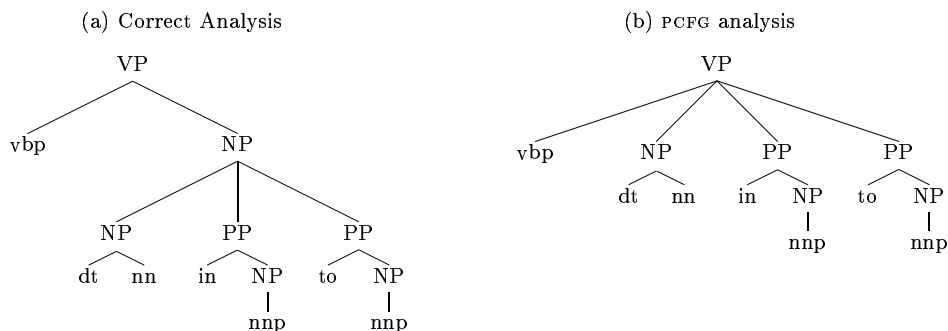
Exact match accuracy on the Wall Street Journal corpus (first line of Tables 3.4 and 3.5) is more problematic: a score of 8.5% and 11.0% respectively was achieved. Whereas in the ATIS-experiment, there was a discrepancy between precision and recall scores, Tables 3.4 and 3.5 show that for the WSJ-experiments they are about the same. F-scores are low at **64.5%** for the xv-experiment and **71.7%** on Section 23.

Apart from parsing accuracy on one particular partition, there were no anomalies in the experimental results, probably because the grammar sparseness problem that was encountered on some partitions of the ATIS corpus are less likely to come into play in a larger-scale corpus such as the WSJ. The low exact match accuracy on Partition 1 is resolved on the other systems and the normal precision and recall scores indicate that no systematic error is causing this abnormally low score. Data analysis showed that the low score can largely be attributed to an unfavorable resolution of ties.

Despite reasonable F-scores for these experiments, serious and fundamental limitations to the PCFG-model can be observed on examining the parsed data. The next example displays the most common type of mistake made by PCFGs. Structure (a) could represent an analysis for the VP-structure in the sentence: *I want a flight from Brussels to Toronto*. (b) represents the analysis the PCFG-model has produced.

Wall Street Journal Corpus (2.21/23)				
Parser	Exact Match (%)	LP	LR	$F_{\beta=1}$ (%)
PCFG	11.0	72.8	70.6	71.7
PMPG	7.3	65.5	64.6	65.0
PCFG+PMPG	16.0	81.8	79.3	80.5

Table 3.5: Experimental Results wsJ(2.21/23) - Overview



This example shows that PCFGs have a tendency to prefer flatter structures over embedded structures. This is a trivial effect of the mathematical formula used to compute the probability of a parse-tree: embedded structures require more rewrite rules, thereby adding more factors to the multiplication, which will almost inevitably result in a lower probability. This claim is further corroborated by the absolute data figures (cf. Appendix B): the absolute number of constituents in the annotated corpus is much higher than the number of constituents the PCFG generates

It is an unfortunate property of PCFGs that the number of nodes in the parse tree is inversely proportionate to its probability. This effect has also been noted by [Magerman and Marcus 1990; Chitrao and Grishman 199; Briscoe and Carroll 1993; Charniak et al. 1996; Bod 2000]. One might be inclined to normalize the probability of a parse tree relative to the number of nodes in that tree, or by introducing the governing properties of the categories in any given context in the model [Johnson 1998]. However, a more linguistically motivated alternative is at hand: the enhancement of context sensitivity through the use of larger syntactic context within parse trees can make our disambiguator more robust.

3.7 PMPG-experiments

An overview of the results for the PMPG-experiments on the ATIS corpus can be found on the second line of Table 3.3. The results are interesting in that the F-score stays about the same, while exact match accuracy is significantly higher. This indicates that more sentences are parsed completely correct. On average, however, the other sentences are parsed slightly worse. Also, while recall scores increase, precision decreases. This means that, on average, a PMPG will propose more constituents, thereby covering more constituents found in parses, but that the overall percentage of the extra constituents it proposes is lower.

The error analysis below indeed shows that PMPG is somewhat of a miss-or-hit affair: it is very good at recognizing patterns in a syntactic structure, but not able to reach DOP-standards when it comes to overall parsing⁹. For the ATIS corpus, PMPG gains significantly on the precision score, but loses out on recall, while these scores remain level on the WSJ corpus. Also note that in contrast to a PCFG, there is not a big difference anymore between the number of constituents found in the annotation and the number of constituents generated by the PMPG. This can be attributed to PMPG's preference for larger substructures, as will be exemplified in the error analysis below.

We recognize the same trend in the 10xv Wall Street Journal experiments (second line of Tables 3.4): again, the F-score does not improve by a very great margin, but the increase in exact match accuracy is quite considerable. There is an increase for all scores on all partitions. This is in contrast to the experiments on the WSJ-corpus with Section 2→21 as the training set and Section 23 as the test set (Tables 3.4 and 3.5). Here a significant decrease on all scores is apparent. Data analysis showed that the overestimation of context size is very apparent for these experiments. The cause of the difference between the two WSJ experiments may be attributed to a problem in robustness for PMPG when increasing the size of the training set.

⁹Note that these results are quite dissimilar from those reported in [De Pauw 2000b] and [De Pauw 2000a], where a decrease of exact match accuracy for PMPG was being observed with respect to the baseline PCFG-model. We believe that the new results are due to the difference in the pre-processing parsing stage, which was less problematic in these experiments as it was for the previous experiments. The reason for this lies in (a) the slightly bigger corpus used in these experiments (578 sentences as opposed to 562 sentences) and can also be attributed to (b) a different distribution of the partitions.

The dynamics in results when moving from a simple PCFG to the more elaborate PMPG calls for an error analysis of the parsed data. Figure 3.15 shows a mistake the PMPG-disambiguator has made during the ATIS-corpus experiments. The correct analysis (a) represents the gold-standard tree-structure for a sentence like *What flights xxx can I get from Brussels to Toronto*. The PMPG analysis (b), featuring a large, nonsensical SBAR-structure, would never have been considered a likely candidate by a common PCFG. This particular sentence was in fact effortlessly disambiguated in the PCFG-experiments. Yet, the fact that large chunks of this tree-structure have been retrieved from memory, makes it the preferred parse for the PMPG.

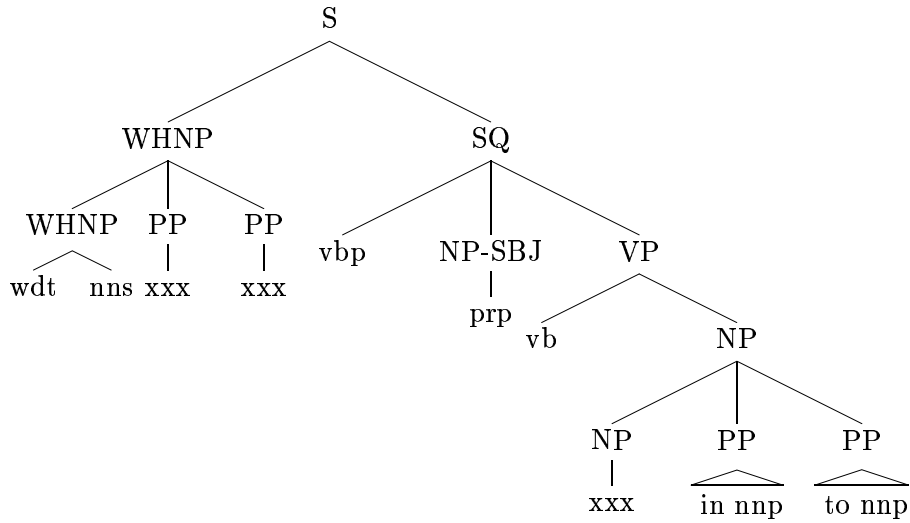
Clearly, PMPG overestimates substructure size as a feature for disambiguation. It is interesting to see however, that it is a working implementation of context sensitivity, eagerly matching patterns from memory. At the same time, it does seem to have lost track of the common-sense probabilistic sensibilities a simple PCFG is able to exhibit. It is in the combination of the two systems that one may find a good disambiguator and accurate implementation of context-sensitivity.

3.8 A Combined System (PMPG+PCFG)

Table 3.3 showed that about 60% of the time, a PCFG finds the correct parse, meaning that the correct parse is at the first place in the ordered parse forest. **99%** of the time, the correct parse can be found among the 10 most probable parses in the ordered parse forest. This opens up plenty of possibilities for optimization. One might for instance use a best-first strategy to generate only the 10 best parses, significantly reducing parsing and disambiguation time. An optimized disambiguator might therefore include a preparatory phase in which a simple PCFG retains the most probable parses, so that a more sophisticated follow-up scheme need not bother with senseless analyses.

In our experiments, we combined the basic sensibilities of a PCFG and used its output as the input for the PMPG. This is a well-established technique usually referred to as *system combination* (e.g. [van Halteren et al. 1998] for an application of this technique to part-of-speech tagging):

(a) Correct Analysis



(b) PMPG Analysis

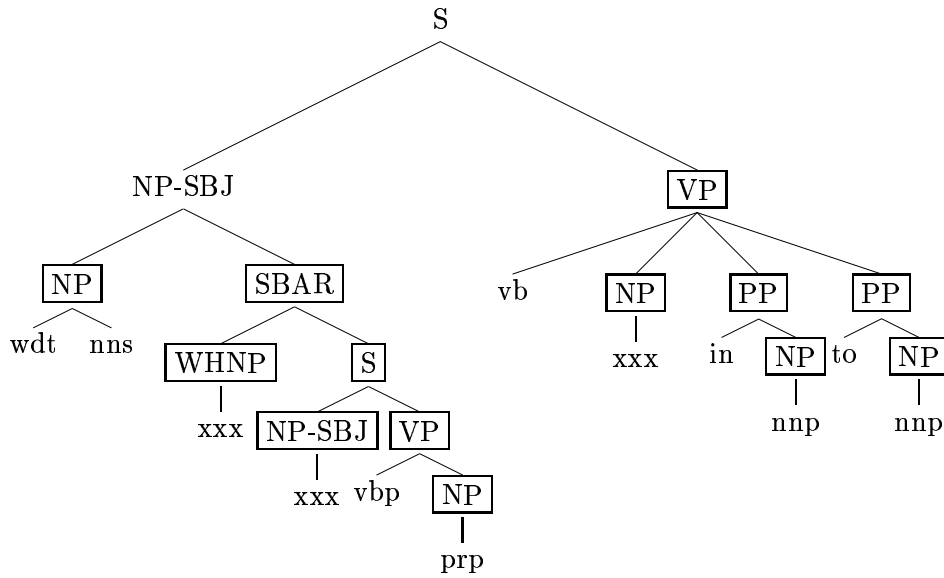
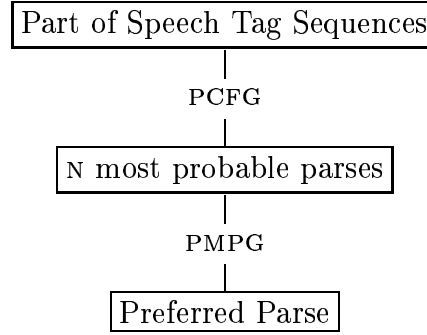


Figure 3.15: PMPG Error Analysis



Sentences are first parsed by a PCFG, that generates the n most probable parses and their initial probabilities (in a viterbi-type fashion). These parses are then presented to the pattern-matching algorithm that adjusts the probabilities of the parses according to their similarity to previously recorded structures (possibly entailing a re-ordering of the parse forest). The combined system then presents the preferred parse.

Since the pattern-matching phase does not precede the probabilistic phase anymore (as was the case in the original PMPG-method), we no longer prune the parse tree. We simply divide the probability that the PCFG assigned to the complete parse tree by the number of non-indexed nodes. A weight can be attributed to each operator's "score", according to the dataset. A homogeneous corpus for instance may benefit from a stronger pattern-matching component.

$$P(\text{parse}) = \frac{\prod_{i=1}^n P(\text{rule}_i)}{\text{number of non indexed nodes}} \quad (3.1)$$

The weight of each algorithm's decision, as well as the number of most probable parses that are extrapolated for the pattern-matching algorithm, are parameters to be optimized.

Experimental Results

The third line in Tables 3.3, 3.4 and 3.5 shows that the combined system performs better than either one of its components, with an exact match accuracy of **74.0%** for the ATIS-corpus and **15 to 16%** for the Wall Street Journal-corpus. F-scores also rise with **90.6%** for the ATIS corpus, **81.9%** on the 10xv WSJ-experiment and up to **80.5%** on Section 23 of the WSJ-corpus, which brings the system in range of the state-of-the art parsers (see Section 3.10).

The parsed data reveals that the combined system is indeed able to overcome difficulties of both algorithms. PCFG+PMPG generates less constituents than the PMPG, but more than a PCFG. This is an indication of the fact, that it is able to overcome PCFG's preference to flat structures, while not indulging in PMPG's careless embedding.

Error Analysis

The parse tree in Figure 3.16 represents a tree-structure for a sentence like *I want a flight from Los Angeles to Miami tomorrow*. The PCFG method provided the parse in Figure 3.17 for this sentence. We notice the aforementioned typical problematic behavior of PCFGs with respect to embedded structures. Figure 3.18 displays the PMPG analysis, with the boxed nodes representing structures retrieved from memory. There is an unnecessary embedding in the first PP. The overall structure of the object NP has nevertheless been reasonably well retrieved.

Figure 3.19 shows the parse for the combined system, which is equal to the correct one. The PCFG's sensibility has made sure that the unnecessary embedding of the NP-structures within the first PP has disappeared, while the PMPG component has gotten rid of PCFG's preference for flatter structures.

There was also a problem in both systems with the constituent structure for *tomorrow*. PCFG as well as PMPG attached the noun to the NP of the second PP. PMPG's mistake can be attributed to the fact that, although rare and thus low in probability, the rule $NP \rightarrow NNP\ NN$ can be found in memory and is accordingly attributed probability 1. PCFG attached the *nn* to the NP, because it prefers trees with lesser non-terminal nodes. But by

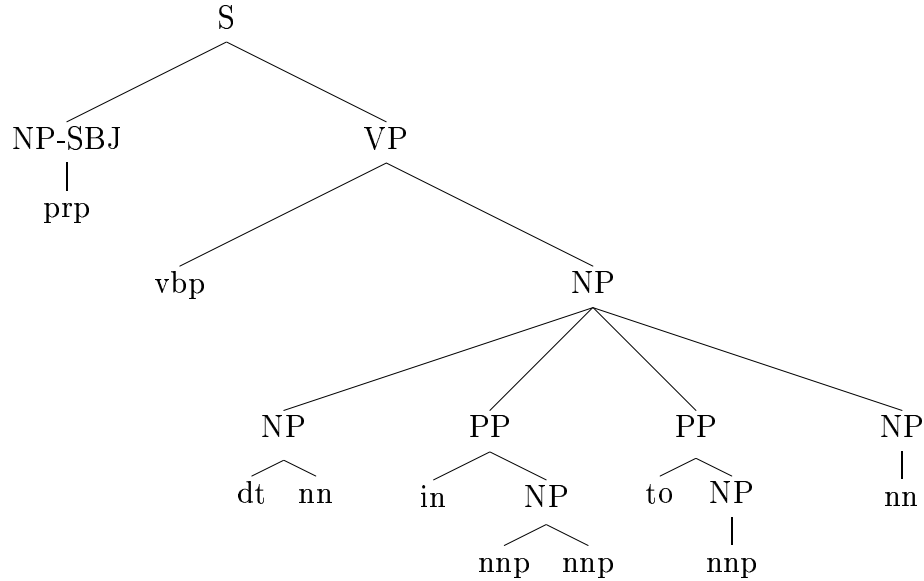


Figure 3.16: Tree-structure for the sentence *I want a flight from Los Angeles to Miami tomorrow*

combining both systems and thus effectively countering PCFGs preference for flatter structures, as well as PMPG over-eager pattern-matching, the correct structure is preferred.

3.9 Comparative Quantitative Data Analysis

This section will provide a comparative overview of the experimental results. The top section of Table 3.6 shows the comparison on the sentence-level. The table needs to be interpreted as follows: out of a total of 578 sentences, 302 were parsed correctly by all three disambiguators alike. 94 sentences were disambiguated incorrectly¹⁰ by all three methods. Of these 94 sentences, the disambiguators still agreed on the parse for 31 sentences, yielding a *agreement* figure of 333. The *C.Disagreement* figure of 182 expresses how many sentences were parsed correctly by one or two, but not three disambiguators.

¹⁰This figure also includes the 35 sentences that could not be parsed correctly by the parsing pre-process.

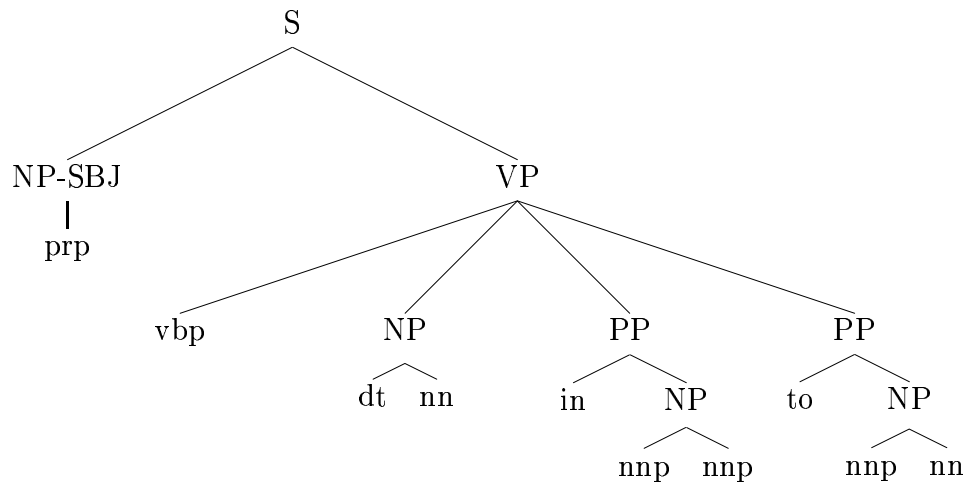


Figure 3.17: PCFG analysis

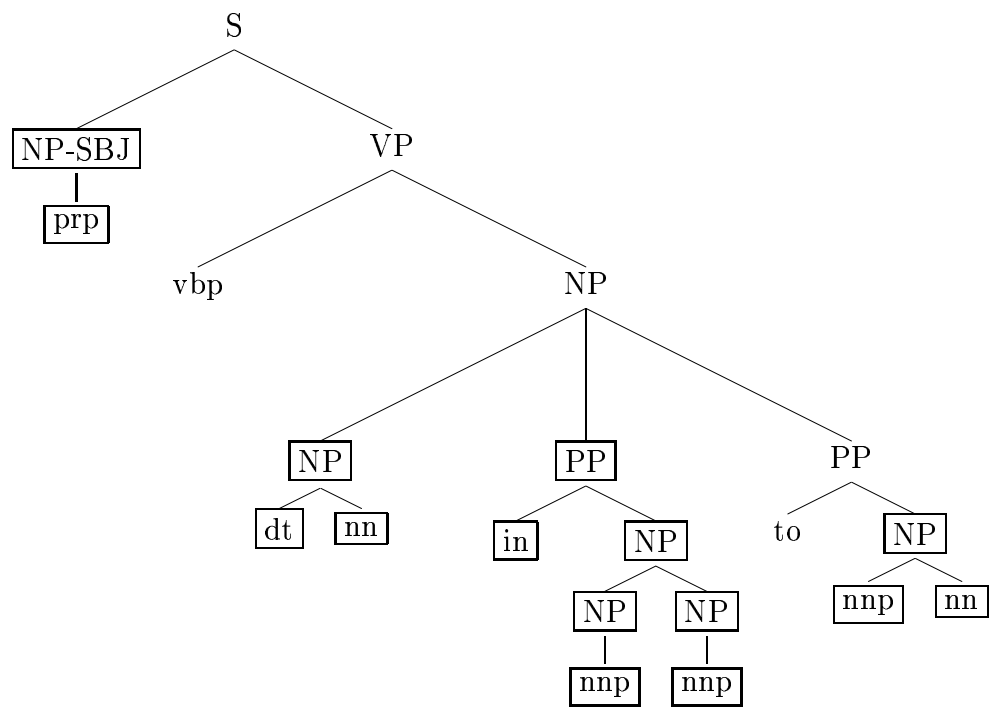


Figure 3.18: PMPG analysis

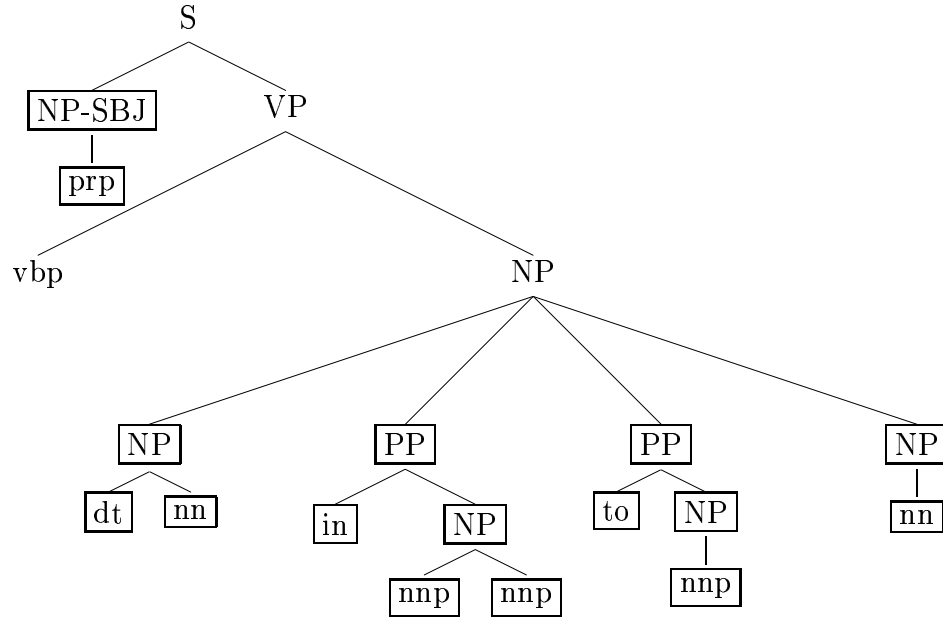


Figure 3.19: PCFG+PMPG analysis

The right-hand side of Table 3.6 displays the distribution of these 182 sentences. PCFG was able to disambiguate 14 sentences that neither PMPG or PCFG+PMPG were able to. PCFG and PMPG were able to parse 5 sentences that the combined system PCFG+PMPG got wrong, while there were 26 sentences that PCFG and PMPG were able to disambiguate, but not PMPG. PMPG was the only one to parse 37 particular sentences. PCFG+PMPG is the clear winner with 49 sentences that could not be parsed by either alternative method. The same comparison is made on the constituent level as well.

Table 3.6 shows that, apart from finding the only correct analysis for 49 sentences, the combined system PCFG+PMPG also incorporates the decisions of its parts: 77 times (26+51) one of the composing parts forces the combined system into the right solution. For 56 sentences, however, the combined system does not have the advantage over PCFG or PMPG.

Accuracy may be further improved if we consider a meta-classifier that chooses among three alternative parses, provided by PCFG, PMPG and PCFG+PMPG respectively. In the best-case scenario, this approach may yield a maximum exact match accuracy of 83.7%. Recall may climb up to a maximum score

Total Sentences	578		PCFG	PMPG	PCFG
Correct by All	302	(52.2%)			PMPG
Wrong by All	94	(16.3%)	PCFG	5	26
Agreement	333	(57.6%)	PMPG	37	51
C.Disagreement	182	(31.5%)	PCFG+PMPG	26	49
Total					
Constituents	4819		PCFG	PMPG	PCFG
Correct by All	3750	(77.8%)			PMPG
Wrong by All	369	(7.7%)	PCFG	29	133
Agreement	4006	(83.1%)	PMPG	116	215
C.Disagreement	700	(14.5%)	PCFG+PMPG	133	173

Table 3.6: Comparative Data Analysis - ATIS

of 92.3% using this approach. Section 3.10 will attempt this by employing a simple weighted voting technique.

We used the McNemar test to calculate the statistical significance of differences in results. The McNemar test studies the difference in two sets of results, only differing from each other in one variable. [Dietterich 1998] suggests it can be used to compare the results of two classifiers to determine whether or not one significantly performs better than the other. [Dietterich 1998] further states that the McNemar test is only valid if the variability in the training set used between algorithms and the internal randomness in the algorithms are limited. Since the training sets in both algorithms are identical and the internal randomness is indeed limited, the McNemar test is a workable approximate heuristic test for calculating statistical significance.

Table 3.7 (reproduced from [Dietterich 1998] and adapted to the parsing domain) displays the way in which statistical significance figures are calculated using the McNemar test. Given the amount of sentences/constituents parsed correctly by method A, but not by method B and vice versa, we are able to calculate the significance of the difference using the formula in Table 3.7. This formula applies a continuity correct of -1 in the numerator, since McNemar's statistic is discrete whereas χ^2 is continuous. We can then use the standard chi-square table to determine the probability that the methods are equal when their results yield the figure returned by the formula.

Table 3.8 displays the McNemar test for the experiments on the ATIS

number of sentences parsed wrong by both methods (n_{00})	number of sentences parsed wrong by method A, but not by method B (n_{01})
number of sentences parsed wrong by method B, but not by method A (n_{10})	number of sentences parsed correctly by both methods (n_{11})

$$P\left(\frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}\right)$$

Table 3.7: McNemar Statistical Significance Test

Methods Compared	Exact Match Accuracy		Recall	
	PCFG vs PMPG	143 40	88 307	540 169
	$P(17.3) < 0.05$		$P(51.8) < 0.05$	
PCFG vs PCFG+PMPG	131 19	100 328	483 65	388 3883
	$P(53.8) < 0.05$		$P(228.9) < 0.05$	
PMPG vs PCFG+PMPG	108 42	75 353	403 145	306 3965
	$P(8.8) < 0.05$		$P(56.8) < 0.05$	

Table 3.8: Significance Tests for experiments on the ATIS corpus

corpus (layout of the tables mirrors Table 3.7). For significance at the 0.05 level, the figure calculated by the formula in Table 3.7 should therefore not be greater than 3.841. For each combination of methods described in this chapter, this figure is much larger than 3.841 on the ATIS corpus, so that the probability that both methods yield the same results is less than 0.05, thereby making the differences in results significant.

Similar results can be observed on the WSJ(xv)-experiments: PCFG+PMPG is the only disambiguator to find the correct analysis for 100 particular sentences, while incorporating PCFG's and PMPG's sensibilities for 111 more sentences. Note that PMPG has a rather high number of sentences (82) that it alone can parse correctly. This may indicate that the weights attached to the systems in the combination (see Section 3.8) need to be optimized. But

Total Sentences	1921			
Correct by All	76	(4.0%)		
Wrong by All	1513	(78.8%)	PCFG	24
Agreement	199	(10.4%)	PMPG	15
C. Disagreement	332	(17.3%)	PCFG+PMPG	48
				63
				100

Total				
Constituents	39098			
Correct by All	20369	(52.1%)		
Wrong by All	5768	(14.8%)	PCFG	234
Agreement	26604	(68.0%)	PMPG	1068
C. Disagreement	12961	(33.2%)	PCFG+PMPG	3465
				542
				3989
				3667

Table 3.9: Comparative Data Analysis - WSJ(xv)

corroborating the intuition that PMPG seems to be biased toward maximizing exact match accuracy, is the low number of constituents it alone has found (542), as opposed to PCFG+PMPGs high number (3667).

Again, assuming we can develop an optimal meta-classifier, we can increase exact match accuracy to 21.2% and a recall score of 81.3%. McNemar tests on these experiments (Table 3.10) indicate that all differences are statistically significant.

The comparative analysis of the WSJ(2→21/23) experiments provides some interesting details. Despite the fact that exact match accuracy drops for PMPG during these experiments, the data analysis shows that PMPG is not performing as badly as the results might indicate: despite a lower overall accuracy, it does come up with 121 analyses for sentences that the other disambiguators are not able to parse correctly, which is impressive in its own right. The cross section with other methods yields relatively low figures (19,13), further proving that in these experiments PMPG performs in a league of its own on these experiments, providing correct analyses for sentences that neither PCFG or PCFG+PMPG can handle, but providing erroneous parses for other sentences.

An optimal meta-classifier can increase exact match accuracy to 24.4% and recall to 82.2%. The McNemar test (Table 3.12) indicates that the differences in results between all models are statistically significant.

Methods Compared	ExMa		Recall	
PCFG vs PMPG	1613	145	9431	4531
	72	91	3699	21437
	$P(23.9) < 0.05$		$P(83.9) < 0.05$	
PCFG vs PCFG+PMPG	1595	163	6306	7656
	39	124	1302	23834
	$P(75.5) < 0.05$		$P(4505.5) < 0.05$	
PMPG vs PCFG+PMPG	1537	148	5998	7132
	97	139	1610	24358
	$P(10.2) < 0.05$		$P(3486.8) < 0.05$	

Table 3.10: Significance Tests for experiments on the wsJ(xv) corpus

Total Sentences	2416			
Correct by All	23	(1.0%)		
Wrong by All	1827	(75.6%)		
Agreement	192	(7.9%)		
C. Disagreement	566	(23.4%)		
Total				
Constituents	47333			
Correct by All	25057	(52.9%)		
Wrong by All	6490	(13.7%)		
Agreement	32044	(67.7%)		
C. Disagreement	15946	(33.4%)		

	PCFG	PMPG	PCFG PMPG
PCFG	62	19	161
PMPG	19	122	13
PCFG+PMPG	161	13	189

	PCFG	PMPG	PCFG PMPG
PCFG	330	1.428	6.598
PMPG	1.428	1.710	2.380
PCFG+PMPG	6.598	2.380	3.500

Table 3.11: Comparative Data Analysis - wsJ(2.21/23)

Methods Compared	ExMa		Recall	
PCFG vs PMPG	2016	135	9830	4090
	223	42	6928	26485
	$P(21.1) < 0.05$		$P(730.5) < 0.05$	
PCFG vs PCFG+PMPG	1949	202	8040	5880
	81	184	1758	31655
	$P(50.9) < 0.05$		$P(2223.4) < 0.05$	
PMPG vs PCFG+PMPG	1889	350	6660	10098
	141	36	3138	27437
	$P(88.1) < 0.05$		$P(3658.8) < 0.05$	

Table 3.12: Significance Tests for experiments on the WSJ(2.21/23) corpus

3.10 Simple Weighted Voting

In the previous section, we provided some accuracy figures an optimal meta-classifier may yield. The McNemar tests (Tables 3.8, 3.10 and 3.12) indicate that there is a very big difference between the three parsing methods, with differences in results being highly significant. Given these often surprisingly disparate decisions made by the three parsers, a majority voting method may provide a significant performance boost. In this section we describe a simple implementation of such an approach for full parsing.

A well-established way to combine the output of different classifiers is to use majority voting: each classifier suggests a class and the class that is most often suggested is the one proposed by the voting method. A refinement of this system attributes a weight to each classifier’s decision, with some classifiers’ vote bearing more on the final decision than others. This approach has been proved quite effective on the part-of-speech tagging task [van Halteren and Daelemans 2001], but it does not readily apply to full parsing. Whereas the number of classes is typically limited for classifiers using propositional data, the number of classes in full parsing is by definition unlimited, rendering the majority voting method moot. We would need a very large number of classifiers to reasonably consider majority voting.

We therefore need to develop an alternate method: rather than looking at the final decision of each classifier, the voting method described in this

section uses the parse forests that each of the parsers provide. The voting method takes the n most probable parses of each parser¹¹, adds up their respective probabilities and returns an ordered miniature parse forest of at most 30 parses. The parse with the highest probability is the parse proposed by the voting method.

If one of the three parsers attributes a particularly high probability to some parse, its high degree of “certainty” about the analysis is reflected in the final decision. This method may even provide correct parses that were previously not considered by any of the parsers: consider the hypothetical situation in which all three parsers have a different parse at the top of their ordered parse forest, yet they all have the same runner-up in the parse forest. The voting mechanism may very well propose this runner-up as the parse of choice, even though none of the individual classifiers had suggested it before.

The results for this method, found in Table 3.13 and 3.14, are quite encouraging with accuracies being improved considerably over the individual classifiers. We observed highly disparate parsing behavior of the individual parsers in the comparative data analysis of the WSJ(2.21/23) experiments and the weighted voting method does indeed significantly boost performance.

The accuracies hold up well in comparison to related work using the same training set and test set division. Table 3.15 reproduced from [Tjong Kim Sang 2002] displays precision and recall scores for the state-of-the-art parsing systems. The results are still considerably lower than those of the state-of-the-art systems, but this can be partly ascribed to (a) the fact that different orthographic edits were conducted for the experiments described in this chapter¹² and (b) the fact that many of the figures in Table 3.15 represent scores on sentences containing less than 40 words (2245 sentences). The scores in Tables 3.13 and 3.14 represent accuracies on all 2416 sentences of Section 23, which includes a fairly large number of unparsable sentences¹³.

¹¹ n was set to 10 in this experiment.

¹²Syntactic flags were not removed and the distinction between ADVP and PRT was maintained. We estimate that exact match accuracy scores especially are affected by this difference.

¹³[Henderson and Brill 2000] describe an experiment in which 39832 parsers were built on the basis of each (and only) sentence in the training set. Experiments showed that 11.2% of these sentences could not be parsed correctly by these parsers.

	PCFG	PMPG	PCFG PMPG	weighted voting
ATIS	60.0	68.3	74.0	78.9
WSJ(xv)	8.5	12.3	14.9	19.1
WSJ(2.21/23)	11.0	7.3	16.0	22.9

Table 3.13: Results for a Simple Weighted Voting Method: Exact Match Accuracy

	PCFG			PMPG			PCFG+PMPG			Weighted Voting		
	LP	LR	$F_{\beta=1}$	LP	LR	$F_{\beta=1}$	LP	LR	$F_{\beta=1}$	LP	LR	$F_{\beta=1}$
ATIS	88.0	81.9	84.8	85.6	85.3	85.4	92.7	88.6	90.6	93.5	90.2	91.8
WSJ(xv)	64.8	64.3	64.5	67.0	66.4	66.7	83.4	80.5	81.9	82.7	83.3	83.0
WSJ(23)	72.8	70.6	71.7	65.5	64.6	65.0	81.8	79.3	80.5	83.0	82.4	82.7

Table 3.14: Results for a Simple Meta-Classifier: Precision/Recall

	LP	LR	$F_{\beta=1}$
[Collins 2000b]	89.9	89.6	89.7
[Bod 2001]	89.7	89.7	89.7
[Charniak 2000]	89.5	89.6	89.5
[Collins 1999]	88.3	88.1	88.2
[Ratnaparkhi 1998]	87.5	86.3	86.9
[Charniak 1997]	86.6	86.7	86.6
[Goodman 1998]	84.8	85.3	85.1
[Magerman 1995]	84.3	84.0	84.1
[Tjong Kim Sang 2001]	82.3	78.7	80.5

Table 3.15: Precision and Recall for Parsing systems on Section 23 of the WSJ corpus

3.11 An integrated system for PCFG+PMPG

All experiments described so far have dealt with disambiguation as a post-parsing process. It is however perfectly possible to integrate both elements of the combined system in one single process. The grammar in Table 3.2 can be employed as a standard grammar in any CFG-parsing system, in the same vein as the method outlined in [Bod 2001]. This way, the parsing system is able to find the best parse in a viterbi-like manner.

Computing the probabilities is done in the same way as for a normal PCFG: the probability of a rule is equal to its relative frequency in the grammar:

$$P(rule) = \frac{|A \rightarrow BC|}{|A \rightarrow \dots|} \quad (3.2)$$

In the same manner as described in Section 3.3.2, the probability of (partly) indexed rules is computed by summing the probabilities of its valid permutations. This method yields equivalent results to the post-parsing disambiguation method with differences in exact match accuracy that are insignificant and due to the random resolution of ties.

Parsing time using this method is 1 second per sentence for the ATIS-corpus, on a PIII 500MHZ linux machine, using a CYK-parser [Chappelier and Rajman 1998]. Wall Street Journal sentences in the 10-fold cross-validation experiments took about 90 seconds per sentence to parse.

Due to these favorable parsing times, this parsing system will be used during most of the experiments described in this dissertation.

Method	Training Phase	Testing Phase	ATIS ExMa
PCFG	$P(rule) = \frac{A \rightarrow BC}{A \rightarrow \dots}$	$P(tree) = \prod_{i=1}^n P(rule_i)$	60.0
PMPG	<ul style="list-style-type: none"> ◊ index tree-structures ◊ $P(rule) = \frac{A \rightarrow BC}{A \rightarrow \dots}$ ◊ store permutations of indexed structures 	<ul style="list-style-type: none"> ◊ index parse ◊ prune parse ◊ $P(tree_{pruned}) = \prod_{i=1}^n P(rule_i)$ ◊ add probability of valid permutations of indexed rules to product 	68.3
PCFG+ PMPG	<ul style="list-style-type: none"> ◊ index tree-structures ◊ $P(rule) = \frac{A \rightarrow BC}{A \rightarrow \dots}$ 	<ul style="list-style-type: none"> ◊ index parse ◊ $P(tree) = \frac{\prod_{i=1}^n P(rule_i)}{\# \text{ of non indexed nodes}}$ 	74.1
PCFG+ PMPG INTE- GRATED	<ul style="list-style-type: none"> ◊ index tree-structures ◊ store permutations of indexed structures in grammar ◊ $P(rule) = \frac{A \rightarrow BC}{A \rightarrow \dots}$ 	◊ $P(tree) = \prod_{i=1}^n P(rule_i)$	74.1

Table 3.16: Comparing four disambiguators

3.12 Summary

Table 3.16 outlines the four disambiguators that were described in this chapter. This table contains the method for training the data, the disambiguation method and the exact match score on the ATIS-corpus.

3.13 Current limitations of the research

Even though the PMPG shows a lot of promise in its exact match accuracy, the following limitations are still apparent:

- the graph in Section 3.8 shows a possible optimized parsing system, in which a pre-processing PCFG generates the n most likely candidates to be extrapolated for the actual disambiguator. In this combined system, each component has been assigned a certain weight: these are parameters to be tweaked for each data set as well. This work should include evaluation on a validation set to retrieve the optimal values for these parameters.
- The bottleneck of the sparse grammar problem prevents us from fully exploiting the disambiguating power of the pattern-matching algorithm. The GRAEL-system described in Part II of this thesis will present a possible solution by using evolutionary techniques to generate, optimize and complement existing, corpus-induced grammars.
- The current indexing process is bottom-up driven only. It would be interesting to provide a choice between bottom-up indexing and top-down indexing, to investigate the effect on parsing accuracy. Also, a fully specifying-index scheme could be added, like the one that is featured in [Goodman 1996], in which an index to a node describes both its upper and lower context.
- Even though it can be argued that part-of-speech tagging is not a task for syntax proper, experiments could be organized to investigate the tagging qualities of a DOP-approach. Also, a direct comparison between the parser described in this chapter and the shallow parsing methods discussed in Chapter 2(p.22) would provide interesting information on the parser as a chunking method.
- Semantic flags and relationships, which have been removed in the annotation for these experiments, could be re-introduced, to see if this kind of extra-syntactic information can be induced from syntactic contextual information only.

- The weighted voting method described in Section 3.10 can also be extended by using a validation set to adjust the weights of each classifier’s decision.

3.14 Conclusions

Even though DOP1 exhibits state-of-the-art parsing behavior, the efficiency of the model is problematic. The introduction of multiple derivations causes a considerable amount of computational overhead. Neither is it clear how the concept of multiple derivations translates to a psycholinguistic context.

This chapter introduced a pattern-matching scheme that tries to explicitly implement the memory-based aspects of Data-Oriented Parsing. This was achieved by disambiguating parse forests by trying to maximize the size of the substructures that can be retrieved from memory. This straightforward memory-based interpretation however yields sub-standard parsing accuracy in an experimental setting. A greedy pattern-matching method that analyzes sentences, strictly by compiling them from the largest chunks of syntactic information recorded in memory, without some kind of probabilistic processing does not seem to be an optimal interpretation of Data-Oriented Parsing. The probabilistic processes that are the focal point of DOP are obviously essential for successful disambiguation. This is further corroborated by the fact that the combination of common-sense probabilities and enhanced context-sensitivity through pattern-matching provides a workable parse forest disambiguator, with accuracies comparable to those of regular DOP.

Finally, we introduced an integrated method that emulates the system combination method by employing the indices representing syntactic context directly in the grammar. This method constitutes a computationally efficient approximation of memory-based syntactic analysis and will be the parser used in the consecutive chapters.

Part II

Data-Driven Experiments in the Evolutionary Computing Paradigm

Divide et Impera

Old Latin Proverb, used by a.o. M. Hurault, Machiavelli and Louis XI.

4

An introduction to GRAEL - **GRAM**mar **Evo**LUtion

This chapter introduces the GRAEL environment, which tries to provide a general computational framework in which grammars for natural language can interact and co-evolve according to principles of evolutionary computing. Motivated from an engineering point of view (Section 4.1), GRAEL provides an environment for grammar optimization and induction, but from a more theoretical point of view (Section 4.2), GRAEL may also help us to understand the dynamics of grammar emergence and evolution over time. The basic concepts of GRAEL will be outlined in Section 4.3 to 4.5 and applied to a practical context in the subsequent chapters.

4.1 Grammar Induction and Optimization - The Engineering Perspective

Data-analysis of the output generated by parsers like the ones described in Chapter 3, usually brings to light fundamental limitations to these corpus-based methods. Even though generally providing a much broader coverage than hand-built grammars, parsers that compile their grammar from a corpus of annotated sentences will still not hold enough grammatical information to provide parses for a large number of sentences, as some rules that are needed to generate the correct tree-structures are not induced from the original corpus¹. But even if there were such a thing as a full-coverage corpus-induced grammar, performance could still be limited by the probabilistic weights attributed to its rules. The GRAEL environment outlined in this section, tries to provide a distributed evolutionary computing method for grammar induction and optimization.

4.1.1 Grammar Sparseness

We touched upon the subject of *grammar sparseness* in Chapter 3 (p. 56). The initial parsing stage proved to be quite problematic for a small corpus such as the ATIS-corpus: 35 out of the 578 sentences (which constitutes 6% of the corpus) could not be parsed, due to sparse grammar problems. Even though ATIS is a fairly homogenous corpus, some very specific and unique structures reside in its structures. This means that if this kind of sentence is featured in the test set, chances are considerably large that the correct grammatical information needed to construct the correct parse tree is not induced from the training set. A large corpus such as the Wall Street Journal corpus seems less prone to this type of grammar sparseness. The 10xv experiments showed only 15 unparsable sentences, which accounts for less than 1% of the sentences.

Note however that the 6% and 1% unparsable sentences on the WSJ and ATIS corpus respectively do not include those sentences for which some (erroneous) parse was found, but for which the correct parse could not have been

¹Also consult [Klein and Manning 2001] for an overview of problems in data-driven parsers caused by the information source.

	Parse Coverage	Structural Consistency
ATIS	94%	89%
WSJ	99%	96%

Table 4.1: Parse Coverage *vs* Structure Consistency

found, since the required rewrite rule needed to construct the correct parse tree for a sentence in the test set, was not featured in the induced grammar². The 1% figure for the WSJ is therefore misleading: [Collins 1999] reports that when using section 2-21 as a training set and section 23 test set, 17.1% of the sentences in the test set require a rule not seen in the training set. Even for a large corpus such as the WSJ, sparse grammar is in fact a serious accuracy bottleneck.

We therefore distinguish between *parse coverage* and *structural consistency*, following the terminology used in the debate between [Goodman 1996] and [Bod 1996]. **Structural consistency** expresses how many times the parser was able to generate the correct parse, i.e. for how many sentences the parser was able to generate a parse forest in which the correct parse was featured. [Black et al. 1993] defines the correctness of a parse in terms of the crossing brackets measure, but we will employ a tighter definition in terms of exact match and labeled precision and recall. The **parse coverage** of a parser expresses how many times the parser was able to generate a parse forest for a sentence. It does however not inform us of the correctness of the analyses found. Table 4.1 displays these scores for the ATIS and the WSJ 10xv-experiments described in Chapter 3.

These figures indicate that deficient structural consistency is a genuine problem for the parsing systems described in Chapter 3, and indeed for any data-driven parser, because it limits the accuracy of the disambiguation methods, even before they have started ranking parses in the parse forests.

For the ATIS-corpus we identified NP-annotation as particular problematic to the overall structural consistency. Highly specific and flat structures like the one in Figure 4.1 are unlikely to be induced from the training set when needed to parse the test set. We therefore need to find a way to create new

²There are such 97 sentences in the ATIS corpus (cf. Chapter 7, p. 215).

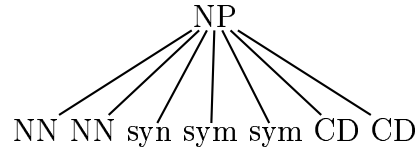


Figure 4.1: NP-structure for *restriction code AP/57*

grammatical information, as well as some kind of filtering mechanism that can distinguish useful grammatical information from noise.

The GRAEL system that is described in this chapter, involves a *distributed* approach to this type of grammar induction. The original (sparse) grammar is distributed among a group of agents, who can randomly *mutate* the grammatical structures they hold. The new grammatical information they create is tried and tested by interacting with each other. The neo-darwinist aspect of this evolutionary system will make sure that any useful mutated grammatical information is retained throughout the population, while noise is filtered out over time. This method provides a way to create new grammatical structures previously unavailable in the corpus, while at the same time evaluating them in a practical context, without the need for an external information source.

4.1.2 Probabilistic Redistribution

In Chapter 3 we mentioned the necessity of tweaking the weights attached to each algorithm's decision in the combined system. Even though the data analysis in Chapter 3 illustrated PCFG+PMPG's ability to overcome parsing accuracies in both components, in many other cases, it can be observed that the ranking of the parse forest is sometimes counter-intuitive in that correct constructs are often overtaken by obviously erroneous, but highly frequent structures.

Table 4.2 displays the comparative data analysis from Chapter 3. This table shows that the PCFG+PMPG disambiguator was able to correctly disambiguate 100 sentences more than the PCFG method and 75 sentences more than the PMPG method. But PCFG and PMPG are able to find the correct

Total Sentences	578			
Correct by All	302	(52.2%)		
Wrong by All	94	(16.3%)		
Agreement	333	(57.6%)		
C. Disagreement	182	(31.5%)		
			PCFG	PMPG
			PCFG	PCFG
			PMPG	PMPG
			PCFG+PMPG	

Table 4.2: Comparative Data Analysis - ATIS (repeated from Chapter 3(p.66))

structure for about 51 sentences that the combined system is not able to disambiguate correctly. This means that for at least 51 sentences, the parts are better disambiguators than the whole. This may indicate that for that particular set of experiments, the weighting of the classifiers in the combined system is not optimal.

One might consider optimizing these weights on a held-out dataset, to consequently use the optimized combined system to disambiguate the sentences in the test set. But the integrated method outlined in Chapter 3 (p. 3.11), does not readily allow such tweaking of the weights, since it is actually an extension of a simple PCFG employing an indexed rewrite-grammar, such as the one we resume in Table 4.3.

The probability of the rules in Table 4.3 are equal to their relative frequency in the grammar. It might be the case however that, even though directly induced from the annotated corpus, the probabilities of these rules are not suited to the disambiguation task as yet. And even though determining the probability of a full parse does entail some extra calculations (Chapter 3 (p. 49)), it may be the case that the distribution of the probability mass over the rules in the grammar does not specifically fit the parsing task yet. It may therefore be useful to have the grammar *practice* the parsing task and adjust the probabilistic weights of particular structures according to these test cases.

Typical methods of probabilistic grammar optimization include, among others, bagging and boosting [Henderson and Brill 2000; Collins 2000b], re-estimation of the constituents probabilities [Goodman 1998; Charniak 2000] and including extra information sources [Belz 2001; Collins 1999]. But the same technique for grammar induction suggested in Section 4.1.1, can be used for grammar optimization as well.

Again, we propose an agent-based evolutionary computing method to resolve this issue. Grammar optimization using a GRAEL environment is in this vein related to the aforementioned bagging approach to grammar optimization, albeit with some notable differences (which will be discussed in more detail in Chapter 6). By distributing the knowledge over a group of agents and having them interact with each other, we basically create a multiple-route model for probabilistic grammar optimization. Grammatical structures extracted from the training corpus, will be present in different quantities and variations throughout the GRAEL society. While the agents interact with each other and in effect practice on each other's grammar, a varied range of probabilistic grammars are optimized in a situation that directly relates to the task at hand. The evolutionary aspects of the system make sure that, while marginally useful grammatical information is down-toned, common constructs are enforced, providing a better balanced model for statistical parsing.

Original	PMPG	
S → NP-SBJ VP	S@12 → NP-SBJ@6 VP@11	S → NP-SBJ@6 VP@11
	S@12 → NP-SBJ VP@11	S → NP-SBJ VP@11
	S@12 → NP-SBJ@6 VP	S → NP-SBJ@6 VP
	S@12 → NP-SBJ VP	S → NP-SBJ VP
NP-SBJ → prp	NP-SBJ@6 → prp	NP-SBJ → prp
VP → vbp NP	VP@11 → vb NP@5 NP@10	VP → vb NP@5 NP@10
	VP@11 → vb NP NP@10	VP → vb NP NP@10
	VP@11 → vb NP@5 NP	VP → vb NP@5 NP
	VP@11 → vb NP NP	VP → vb NP NP
NP → prp	NP@5 → prp	NP → prp
NP → NP PP PP NP	NP@10 → NP@4 PP@9 PP@8 NP@3	NP → NP@4 PP@9 PP@8 NP@3
	NP@10 → NP@4 PP@9 PP@8 NP	NP → NP@4 PP@9 PP@8 NP
	NP@10 → NP@4 PP@9 PP NP@3	NP → NP@4 PP@9 PP NP@3
	NP@10 → NP@4 PP@9 PP NP	NP → NP@4 PP@9 PP NP
	NP@10 → NP@4 PP PP@8 NP@3	NP → NP@4 PP PP@8 NP@3
	NP@10 → NP@4 PP PP@8 NP	NP → NP@4 PP PP@8 NP
	NP@10 → NP@4 PP PP NP@3	NP → NP@4 PP PP NP@3
	NP@10 → NP@4 PP PP NP	NP → NP@4 PP PP NP
	NP@10 → NP PP@9 PP@8 NP@3	NP → NP PP@9 PP@8 NP@3
	NP@10 → NP PP@9 PP@8 NP	NP → NP PP@9 PP@8 NP
	NP@10 → NP PP@9 PP NP@3	NP → NP PP@9 PP NP@3
	NP@10 → NP PP@9 PP NP	NP → NP PP@9 PP NP
	NP@10 → NP PP PP@8 NP@3	NP → NP PP PP@8 NP@3
	NP@10 → NP PP PP@8 NP	NP → NP PP PP@8 NP
	NP@10 → NP PP PP NP@3	NP → NP PP PP NP@3
	NP@10 → NP PP PP NP	NP → NP PP PP NP
NP → dt nns	NP@4 → dt nns	NP → dt nns
PP → in NP	PP@9 → in NP@7	PP → in NP@7
	PP@9 → in NP	PP → in NP
PP → to NP	PP@8 → to NP@2	PP → to NP@2
	PP@8 → to NP	PP → to NP
NP → jj nnp	NP@3 → jj nnp	NP → jj nnp
NP → NP NP	NP@7 → NP@2 NP@1	NP → NP@2 NP@1
	NP@7 → NP@2 NP@1	NP → NP@2 NP@1
	NP@7 → NP@2 NP	NP → NP@2 NP
	NP@7 → NP NP@1	NP → NP NP@1
	NP@7 → NP NP	NP → NP NP
NP → nnp nnp	NP@1 → nnp nnp	NP → nnp nnp
NP → nnp	NP@2 → nnp	NP → nnp
NP → nnp nnp	NP@1 → nnp nnp	NP → nnp nnp

Table 4.3: Extended Grammar (repeated from Chapter 3 (p. 47))

4.2 Grammar Induction and Optimization - The Evolution of Language Perspective

At the same time, theoretical insights can be gained on the development of compositional language in a simulated multi-agent setting. The object of the experiments is to show that computational agents can develop grammar in a self-reliant manner, without the need for an external information source. From an engineering point-of-view, the GRAEL environment described in this chapter, offers a grammar optimization and induction method that can help alleviate problems of grammar coverage and sparse data, as well as optimize the distribution of the probability mass. But by looking at the nature of the interaction between the agents, a more theoretical account of grammar optimization in an evolutionary environment can be given.

The evolutionary computing paradigm has always seemed reluctant to deal with issues of natural language syntax. The fact that syntax is in essence a recursive system, dealing with complex issues such as long-distance dependencies and constraints, has made it difficult to incorporate it in typically propositional evolutionary systems such as genetic algorithms. With the notable exception of Losee's LUST system [Losee 1995] most GA syntactic research has focused on non-linguistic data. [Smith and Witten 1996], [Wyard 1991], [Antonisse 1991] and [Blasband 1998] are some examples on how Genetic Programming can be applied to the induction and optimization of simple artificial grammars. But most of these systems are not suited to investigate the dynamics of grammar evolution itself, mainly because the grammatical system and evolutionary processes underlying these systems are designed to fit a particular task, such as information retrieval [Losee 1995].

Computational simulations of the origins of syntax however, can provide a more relevant account of grammar evolution. [Batali 1998b] implements a simple agent-based architecture, in which a population of agents communicate meaning to each other. Through an example-based learning process agents are able to detect systematic regularities in their communication, and are able to create novel string-meaning pairs. Even without a presupposed Language Acquisition Device [Briscoe 1997], the community of agents is able to achieve a high degree of coordination in their communication. [Kirby 1999] implements a similar agent-based system, in which compositionality is

considered to be an emergent adaptation, necessary for the survival of the language over different generations.

These systems, which we will discuss in more detail in Chapter 9, provide a very interesting account on how syntax may have originated in primitive hominids without presupposing any prior linguistic knowledge. Yet, these systems have a tendency to converge on a particular grammatical model. For the most part, this is due to the relatively limited meanings that the compositional models are able to express, so that convergence is almost inevitable and further language change consequently problematic. This poses a problem when trying to explain grammatical change and evolution over time, which seems to be paramount for the creation of new meaning in particular and language evolution in general. Moreover, the agents in these systems are often attributed an implicit linguistic bias towards compositionality and often establish an unrealistic communication model in which meaning is explicitly shared. We will go into these issues in some more detail in Chapter 9.

The GRAEL environment is also geared toward the goal of investigating the behavior of grammars for natural language in an evolutionary context. In its full form, GRAEL contains a population of agents in a virtual environment. Agents are capable of expressing beliefs about this environment in different ways. Comparable to the systems of [Batali 1998b] and [Kirby 1999], the agents are forced into a converging grammatical system, which will consequently evolve over further generations and adapt to the changing population and environment, avoiding finite convergence.

Different experiments will be described that focus on different aspects of this evolutionary model. And even though the data-driven experiments outlined in Part II are scarcely connected to the aforementioned research on the origins of compositional language, a theoretical account of grammar in an evolutionary context can answer some interesting questions about the behavior of grammatical systems in an evolutionary context:

- What makes for an accurate grammar in an evolutionary context
- What is the best method to transport grammatical information over generations
- How does convergence in a society of agents affect their performance

- What fitness functions yield the best grammars in agents
- ...

4.3 Outline of the GRAEL-system

This chapter describes an implementation of an agent-based evolutionary computing method for grammars, called GRAEL³. Before we go into the details, it is necessary to define some keywords:

The GRAEL system, method Refers to the technicalities of GRAEL. Anything to do with the algorithmic details of GRAEL can be referred to as belonging to the GRAEL system

The GRAEL environment This term is used to refer to the GRAEL approach of using a distributed evolutionary approach to grammar optimization

A GRAEL society A particular group of agents performing a particular task, such as grammar optimization, induction, ...

4.3.1 Architecture

A GRAEL environment typically contains a population of agents in a virtual setting. Before any interaction occurs, grammatical information is randomly distributed over the agents. Next, the agents will engage in a series of interactions, called *language games*, a term borrowed from AI-research investigating the origins of grammar [Steels 1997] and adapted to the concepts of the GRAEL environment. During these games, agents have to agree on some kind of syntactic structure for a sentence. By updating their own information

³GRAEL is an acronym for GRAMMAR EVOLUTION. All GRAEL systems described in this dissertation feature some kind of grammatical system, evolving from an initial state to a state in which it has been optimized for a particular task.

source through practical use, useful grammatical information is re-enforced throughout the environment.

“*Knowledge*”

The grammatical knowledge attributed to agents in the data-driven GRAEL instantiations of Part II is largely determined from an engineering point of view, as we try to provide the agents with a mental structure that will allow them to optimize (and induce) grammars. An agent’s internal knowledge management does therefore not necessarily reflect a psycholinguistically motivated reality.

Each agent’s “mind” contains a number of grammatical structures, constituting analyses for a set of sentences. These structures can either be drawn directly from an annotated treebank (Chapters 5 to 7) or be induced in an unsupervised manner (Chapter 8). The language used by the agents can be pre-defined (Chapters 5 to 8), but a set of experiments will also be described which tries to model the emergence of grammatical language (Chapter 10) within the GRAEL environment.

To illustrate how the GRAEL system operates, we will describe the environment used in Chapter 5. In these experiments, tree-structures of an annotated treebank, i.e. the ATIS and WSJ corpus [Marcus et al. 1993], are randomly distributed among the agents in a GRAEL society. Each agent now holds a limited number of tree-structures in memory, or in other words: each agent has some knowledge about the domain at hand. Figure 4.2 displays the extraction of a GRAEL society from a treebank.

The grammatical information in the agent’s mind is divided into two functional units: an *I-language*⁴ and an *E-language*⁵. These terms were borrowed from Chomskian terminology, but are used in a slightly different sense here. The E-language (“externalized language”) consists of the original tree-structures that the agent received at the onset of the GRAEL-society. One can

⁴Chomsky’s term from the mid-1980s for the knowledge of language internalized by individual speakers. [Matthews 1997].

⁵Chomsky’s term in the mid-1980s for a language conceived as a system of events or utterances or other units external to or as externalized by the individual speaker. Applied, in effect, to all conceptions of language other than as I-language. [Matthews 1997].

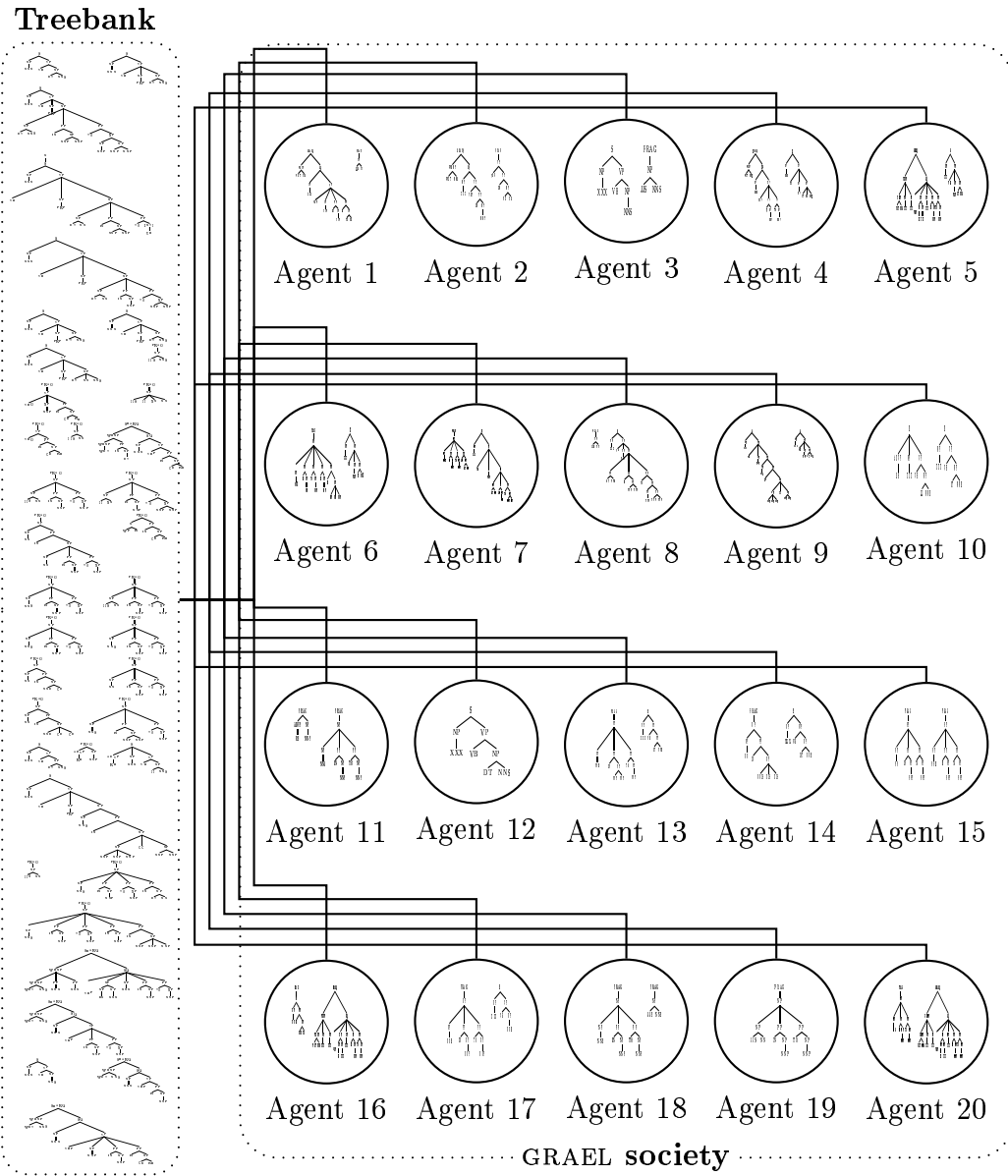


Figure 4.2: Extracting a GRAEL society from a treebank

consider these as sentences the agent in question is able to produce and the mental interpretation the agent has of this sentence. The E-language makes it possible for agents to produce sentences. The agents also induce a grammar⁶ from these tree-structures. These rules constitute the initial I-language (“internalized language”).

By interacting with other agents, the I-language is enriched with new grammatical information. Apart from the initial induction of the I-language from the E-language, there is no interaction between these two components in the experiments described in Chapter 5: while the I-language is updated through language games, the structures in the E-language are not altered. Chapter 5 will explain why it is necessary to deny this interaction for grammar optimization, but for the induction of new grammatical information, we need to allow some limited interaction between the two components. The experiments in Part III however, will abandon the distinction between I-language and E-language, as psycholinguistic relevance becomes more essential to the nature of these experiments themselves.

An Example

Let us take a look at a toy example of a language game, to introduce the operations that underly the communication between agents. Figure 4.3 shows an annotated corpus of two tree-structures, distributed over two agents. The tree-structures become the E-language, from which an initial grammar is induced to create the I-language. Next, the agents in this GRAEL society will start playing language games. Two agents are randomly selected from the society. **Agent2** is the listener, who will try to parse **Agent1**'s sentences⁷. Once **Agent2**'s analysis reaches a certain threshold of parsing accuracy, the language game is considered a success. Parsing accuracy is defined in terms of the F-measure (see Chapter 3 (p.56)). The threshold value is typically set to 1 (exact match accuracy), but a looser definition of communicative success can be used by lowering the threshold value.

In this language game **Agent1** presents the (string-only) sentence *Winslow*

⁶Represented as PMPG-grammar for most of the experiments.

⁷The sharing of knowledge is typically unidirectional: **Agent1** does not learn any new grammatical data from **Agent2**.

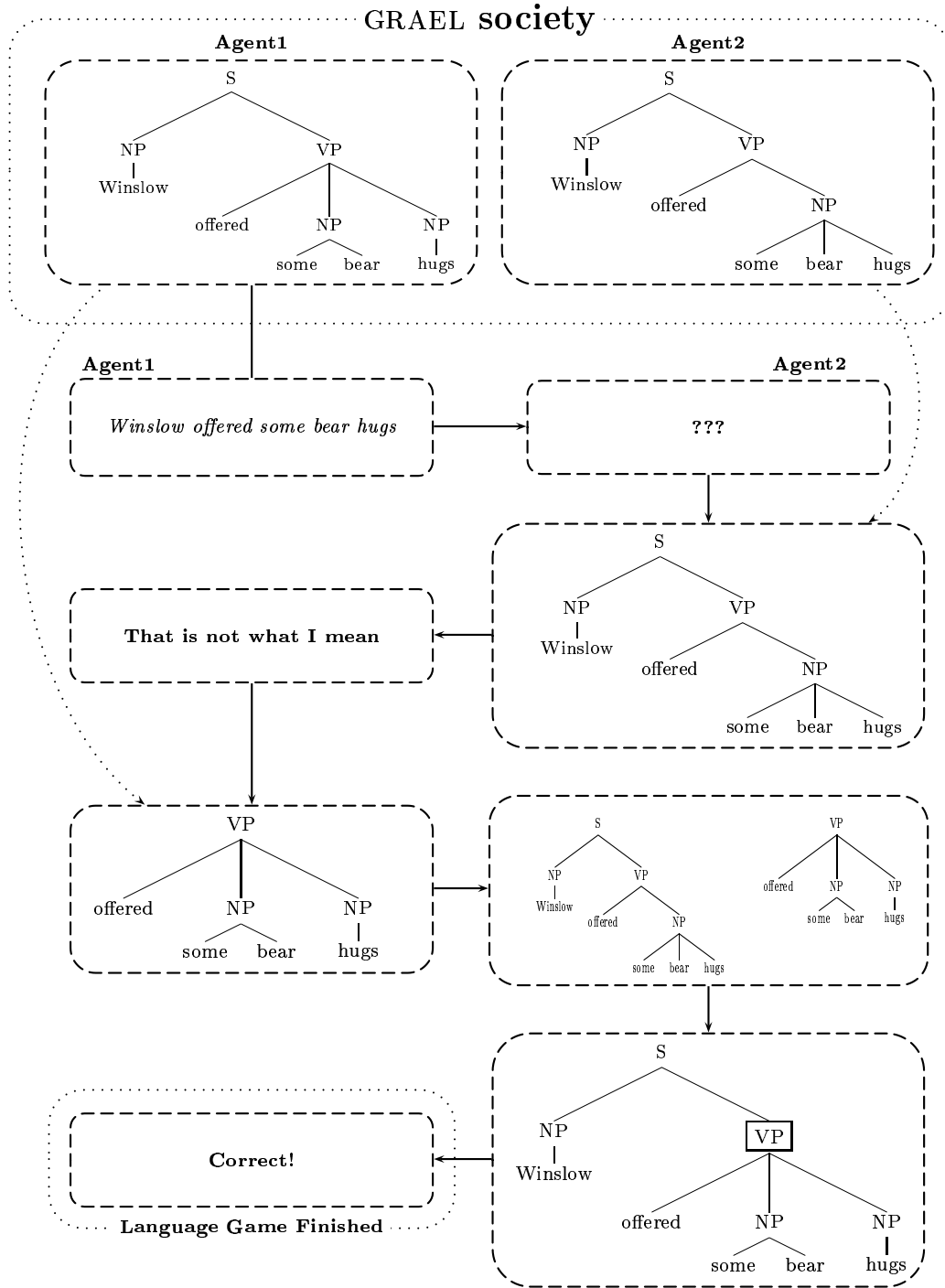


Figure 4.3: A Language Game in a toy GRAEL Society

offered some bear hugs to **Agent2**. **Agent2** tries to parse this sentence and presents a parse which is consistent with his grammar. This analysis is not the one **Agent1** intended, who will consequently try and help **Agent2** out by revealing the possibly relevant correct substructure to **Agent2** in a bottom-up fashion (cf. Section 4.5). **Agent2** will now parse the sentence *Winslow offered some hugs* again with the new grammar recompiled from his updated treebank. **Agent2**'s new analysis is consistent with **Agent1**'s and the language game is a success.

Note that knowledge is explicitly shared, which does not correspond with the original notion of language games, as outlined in [Steels 1997] (also see Chapter 9), in which the success of a language game is typically determined by whether or not the agents indicate the same point of reference in a communicative attempt. Knowledge in these types of language games is not transferred from one agent to another in raw form, as opposed to the explicit knowledge sharing in the GRAEL-societies described in Chapters 5 to 6. The GRAEL approach is justifiable if we are not interested in modeling human-like communication⁸, but in providing a workable method for agent-based grammar optimization and induction.

4.3.2 The Parser

The parser that the agents employ in the corpus-based GRAEL experiments is a memory-based optimization of the DOP-parser (Chapter 3), in which a parse-tree is evaluated in terms of its similarity to previously observed tree-structures in memory. We mostly use the integrated parser, described in Chapter 3 (p. 76), unless computational problems force us to do otherwise.

4.3.3 Different Instantiations of GRAEL

In parts II and III four instantiations of the GRAEL system will be discussed, each focusing on a different aspect or goal. GRAEL-1 (Chapters 5 and 6)

⁸GRAEL-4 (Chapter 10) will try to provide a possible explanation for the emergence of grammar in early hominids and will therefore abandon such direct sharing of knowledge to constitute a more realistic model.

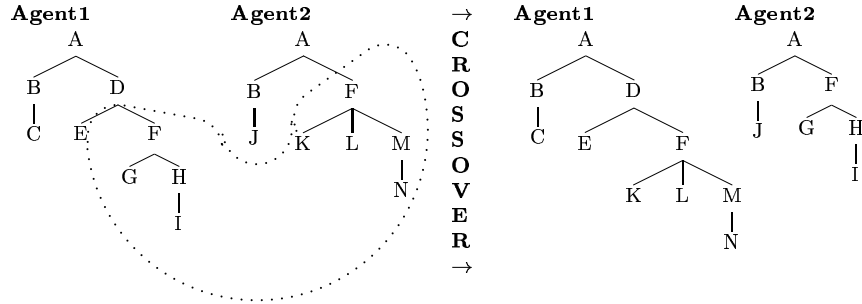


Figure 4.4: Crossover Operation

will deal with grammar optimization and the simple redistribution of grammatical information throughout the GRAEL society. **GRAEL-2** (Chapter 7) will allow the agents in the society to introduce new grammatical information by mutating existing grammatical structures. **GRAEL-3** (Chapter 8) is an unsupervised approach to grammar induction for natural language, while **GRAEL-4** (Chapter 10) will try to provide a possible model for the emergence of an artificial compositional language. The further research moves towards GRAEL-4, the more we will abandon the pre-defined information sources that are available to the agents, as less cognitive capacities are presupposed in the agents. Notions of explicit knowledge sharing, necessary for efficient data-driven grammar optimization, will need to be toned down as we approach GRAEL-4 in favor of a more psycholinguistically motivated model of human-like communication.

4.4 Link with Genetic Algorithms

There are two major operations possible in the GRAEL environment: crossover and mutation. These are classic operators for genetic algorithms, but are used in the GRAEL system in a slightly different sense. **Crossover** is used to describe the operation in which parts of syntactic trees are being exchanged. Figure 4.4 shows a basic crossover operation for two agents. The nature of the crossover operation is inspired by Data-Oriented Parsing (Chapter 3) with its emphasis on the substitution of entire substructures, rather than simple rewrite rules.

There are three types of crossover in the GRAEL environment. Experi-

mental results will show that crossover operation (2) does not improve on the accuracy of agents in a GRAEL society, but it is included for completion's sake:

1. Language Game Crossover This is the default crossover operation in the GRAEL environment. It always occurs in the context of a language game, when **Agent1** reveals part of the correct structure to **Agent2**. The latter will incorporate this new information in his grammar. LG-crossover is limited to be unidirectional, like in the example in Figure 4.3. **Agent1** acts as a teacher to **Agent2** and does not receive any grammatical information. Bidirectional Language Game Crossover would allow **Agent2** to introduce noise in the GRAEL society, degrading overall performance.

2. Random Crossover

An agent can also choose another agent for random crossover. This means that the two agents perform random crossover operations on each other's tree-structures, extending each other's grammar, without actually testing out the grammatical information in a language game. This extends grammar faster than Language Game Crossover, but also introduces more noise. In principle, this type of crossover could be used to speed up knowledge transmission throughout a society.

3. End-of-Life Crossover

Generations (cf. *infra*) allow a GRAEL society to rid itself of badly performing agents, while good agents will live on in their offspring. End-of-Life Crossover occurs when two (fit) agents are selected to crossover their grammatical information on a full-sentence level. In this crossover operation, only full tree-structures are crossed over, not substructures present in the grammar. The resulting agents will constitute members of the next generation in the GRAEL society.

The crossover operation can also be extended to include crossover on nodes with different category labels, in which case it is actually an instance of mutation (cf *infra*). This kind of crossover only occurs during random crossover. An NP node can for example be crossed over with a VP node, although experiments will show that hardly any useful grammatical information can be inferred from relaxing the crossover constraints in this way.

Another classic GA-operator is mutation. Inspired by Darwinist biology, mutation involves a possibly beneficial distortion of data. We define three different types of mutation in the GRAEL environment:

1. Crossover Mutation

occurs when crossover constraints are relaxed in the context of random crossover (cf *supra*).

2. Internal Mutation

an agent can mutate his own grammatical information by randomly adding nodes, deleting nodes or changing node labels (See Figure 4.5. This typically occurs prior to procreation).

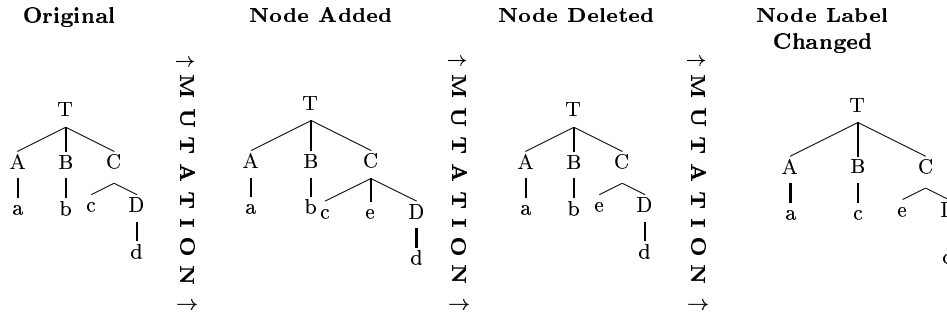


Figure 4.5: 3 different Mutation Operations

3. Noisy Channel Mutation

only occurs in the context of a language game. When **Agent1** suggests a substructure to **Agent2**, a virtual noisy channel may cause **Agent2** to misunderstand the information sent by **Agent1**. This kind of mutation may involve adding nodes, deleting nodes or changing node labels (See Figure 4.5). We will discuss these operations in more detail in Chapter 7.

We have defined six different operations: language game crossover, random crossover, end-of-life crossover, crossover mutation, internal mutation, noisy channel mutation. Of these, only language game crossover is paramount to the GRAEL system.

Generations

Even though not necessary to have a working GRAEL environment, the implementation of generations in a GRAEL society is desirable from a computational point of view. Since crossover-operations as well as mutation create new syntactic structures, an agent's knowledge can grow unwieldy in a short period of time. We therefore impose an implicit upper-limit on the number of grammatical structures an agent can contain, mirroring human memory restrictions.

When an agent has reached this end-of-life threshold, its performance in

the GRAEL society will be evaluated in terms of fitness functions (cf infra). A fit agent's "genetic material" will proceed to a next generation, while an unfit agent will merely disappear from the society.

There are two basic methods of procreation in the GRAEL environment:

- **Sexual Procreation:** The fit agent is allowed to look for another, preferably fit end-of-life agent and perform end-of-life crossover. The number of agents in its offspring is dependent on the parents' size. If it cannot find another end-of-life agent or its performance has been sub-par, it will simply disappear from the society.

- **Asexual Procreation:** New generations can also be created without an end-of-life crossover operation: a fit end-of-life agent can just splice itself into two or more new agents, while unfit agents will disappear from the GRAEL society.

Fitness

Neo-Darwinism is introduced by influencing the above choice between offspring or annihilation on the basis of the success of his communicative attempts in the society. Successful communication can be defined in different ways, by imposing a diverse set of fitness functions on the development of a society:

Efficiency	The computational efficiency of an agent, i.e. the average CPU-time an agent needs to parse a sentence.
Size	The size of an agent's grammar, i.e. the number of rules in the grammar induced from the agent's knowledge. Related to efficiency.
Accuracy	The accuracy with which an agent is able to parse a held-out set of test strings.
Understanding	The average accuracy with which an agent is able to parse the other agent's sentences during the language games.
Understandability	The average accuracy with which other agents are able to parse the agent's sentences during the language games.
Internal Consistency	The accuracy with which an agent is able to parse his own sentences.

The fitness of an agent is a weighted combination of these metrics, so that the grammatical properties of the fittest agent are directly influenced by the weights assigned to each operator. Chapter 5 will describe experiments in which GRAEL societies were optimized for each of these fitness operators. Even though observed differences are often slight, some interesting tendencies will be noticeable.

4.5 The Basic GRAEL Algorithms

The following algorithm outlines the basic GRAEL system, using only language game crossover as an operator.

1. Initialize GRAEL-society
 Threshold U : communicative success (F-score)
 n agents containing tree-structures for k sentences
2. Pick 2 agents
Agent1: holds structures $S_1 \dots S_{k-1}, S_k$ in treebank T_{agent1}
Agent2: compile grammar G from treebank T_{agent2}
3. For each $S_i \in \{S_1 \dots S_{k-1}, S_k\}$
 - (a) **Agent1** displays part-of-speech tag sequence for S_i
 - (b) **Agent2** proposes parse P for S_i consistent with G
 - (c) Calculate F-score F for P
 - If** $F \geq U$
 - then** Go to (3a) with S_{i+1}
 - else** **Agent1** reveals the minimal correct substructure $X \subseteq S_i$
Agent2 adds X to T_{agent2}
Agent2 recompiles G from treebank T_{agent2}
 Go to (3b) (only 3 passes allowed)
4. Go to 2

The next algorithm describes the way **Agent1** determines the minimal correct substructure (cf. Step 3(c) in the Language Game Algorithm).

1. Check for Constituent list CL_1
if CL_1 exists
then go to 4
else go to 2
2. Create word/position list for sentence S_i
"0 w₁ 1 w₂ 2 ... w_n N"
3. Create constituent list CL_1 for **Agent1**'s analysis

0	POS_j	CAT ₁ → ...
POS_{j+1}	POS_{j+k}	CAT ₂ → ...
N-1	N	CAT _m → ...

4. Create constituent list CL_2 for **Agent2**'s parse
5. Find constituents in CL_1 not present in CL_2
6. Traverse CL_1
if no constituent marked
then choose constituent with highest number of non-terminals on right-hand side
mark this constituent in CL_1
else look for previously marked constituent and choose the superordinate node
7. Return marked constituent in CL_1 and its subordinate clauses

4.6 Concluding Remarks

This chapter introduced the GRAEL system, an agent-based evolutionary approach to grammar induction and optimization. While the distributed nature of GRAEL allows for several alternatives to be developed simultaneously, the evolutionary computing aspects make sure that only the best candidates survive over time.

The basic scope of GRAEL is to implement a framework in which sub-standard grammars can improve themselves without the help of an external information source, but simply by interacting with each other, sharing information and practicing a particular task on each other. The following chapters will describe experiments that show that GRAEL can be used to

- adapt a grammar to a very specific purpose, be it grammatical efficiency, accuracy or purely successful inter-agent communication.
- improve a grammatical system by interacting with other (faulty) grammatical systems.
- yield a grammatical system that is superior to the overall grammatical system present in the original corpus at startup time.
- sustain a generation-based environment without finite convergence.

More theoretically, we will look at the evolution and intricacies of each agent's syntactic component for particular (combinations of) fitness-functions.

From an engineering point of view, GRAEL shows that real-world natural language grammars can adaptively improve themselves in an evolutionary context without the need for external manual or automatic statistical and grammatical smoothing.

It's survival of the fittest, Max, and we've got the fucking gun!

Lenny Meyer - Pi - ©1998

5

GRAEL-1 - An Agent-Based Evolutionary Computing Approach to Probabilistic Grammar Optimization

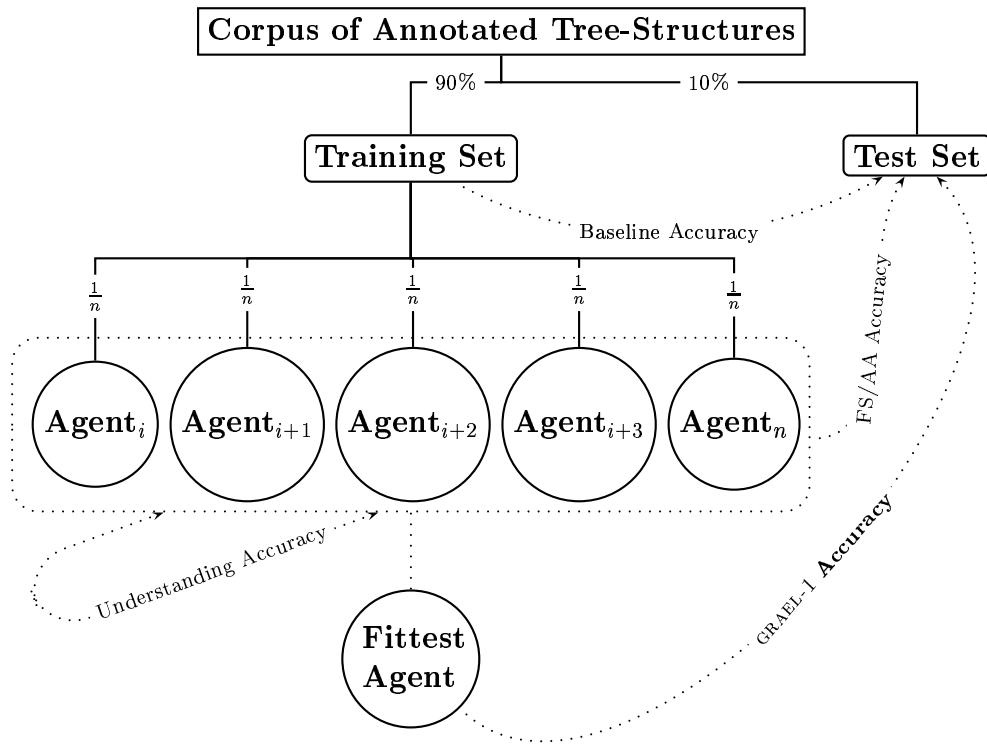
This chapter will discuss the first and most prominent batch of experiments on the GRAEL environment. GRAEL-1 is a data-driven GRAEL-environment, meaning that the grammatical structures being processed in the society are culled from an annotated corpus of tree-structures. The goal of GRAEL-1 is to optimize treebank grammars through an extended series of language games, in which grammatical information is shared, i.e. crossed over between agents. This means that no new grammatical information is created in a GRAEL-1 society at any point: language games only re-distribute the grammatical information that was present in the original corpus, so that the probability mass is re-adjusted over the grammatical elements in a setting that resembles the actual task at hand: parsing unseen data. The overview in Chapter 1 (p. 8) defined the memory-based parsing system in Chapter 3 (PMPG) as a fully specified parsing system: it processed a pre-defined natural language grammar and induced the distribution of the probability mass

directly from the data. GRAEL-1 takes a first step in discarding the pre-defined elements in a parsing system, by implementing a way to dynamically alter the distribution of the probability mass in an agent-based evolutionary computing setting.

In Chapter 4 (pp 4.1,4.2) we defined two distinct uses for the GRAEL system: grammar induction/optimization and the investigation of the dynamics of grammar in an evolving system. GRAEL-1 is more geared towards the former and serves as a grammar optimization method, in that it optimizes the probabilistic distribution of the grammatical chunks in a practical situation. In a sense, the agents in the GRAEL society are *practicing* on each other by performing the task that they are supposed to do. This is reminiscent of an established machine learning optimization technique of using a *validation set*¹ to tweak the training data before it is used to disambiguate a held-out test set. GRAEL is different in that the agents start with inadequate training data, using each other as ad-hoc validation sets.

Using an existing natural language and a pre-defined grammar, the experiments with GRAEL-1 cannot provide a possible explanation for the origins of compositional language in the way that [Batali 1998b; Steels 1997; Kirby 1999] do and GRAEL-4 will attempt to do. Some theoretical insight with respect to the dynamics of grammars in an evolutionary setting can nevertheless be gained by looking at the data generated in the GRAEL society. The pre-parsed natural language corpora provide an objective touchstone on which we can evaluate the grammars developed by GRAEL-1. Furthermore, by alternating the fitness functions in the GRAEL society, it is possible to look at the nature of grammars, which were “genetically” enhanced for some kind of task, such as parsing accuracy or efficiency.

We will discuss the setup of the GRAEL-1 experiments in Section 5.1. The experimental results on the ATIS and WSJ corpus will be presented in Sections 5.2 and 5.3 respectively, after which we conclude with some summarizing thoughts in Section 5.4.



Baseline Accuracy	F-score achieved by the baseline model on the test set
Understanding Accuracy	F-score achieved during language games
FS Accuracy	Full Society Accuracy. The F-score on the test set achieved by a grammar compiled from all agents in a society
AA Accuracy	Average Agent Accuracy. The average F-score achieved by agents' grammars on the test set
GRAEL-1 Accuracy	F-score achieved by the fittest agent's grammar on the test set

Figure 5.1: Default Setup for GRAEL-1 Experiments

5.1 Experimental Setup

Chapter 4 introduced different operations the agents are able to perform in the GRAEL environment. We will deal with the mutation-operation in Chapter 7, but the different types of crossover we defined in Chapter 4, as well as population size and the use of generations will be tweakable parameters in the experiments with GRAEL-1.

5.1.1 General Setup

Figure 5.1 displays the main setup for GRAEL-1 experiments (without a validation set). An annotated corpus of tree-structures is randomly divided into a 90% training set and a 10% test set. The training set is used to parse the test set, like in the experiments of Chapter 3. The accuracy achieved by the grammar induced from the training set will be considered as the *baseline accuracy*. The tree-structures of the training set are then equally divided over the agents in the GRAEL society.

We can measure the *full society accuracy*, by which we mean the accuracy of a parser, powered by a grammar induced from all agents in the society. In the initial stage, when no language games have been played, full society accuracy is by definition equal to the baseline accuracy². Another measure is the *average agent accuracy*, indicating the average accuracy of the agents on the test set. Since the latter involves a lot of computational overhead with little information to be gained, it is usually not measured in the GRAEL-1 experiments.

Chapter 4 (p. 102) also described some fitness functions. The understanding fitness function for instance measures the average accuracy with which an agent is able to parse another agents' sentences. We can determine the average *understanding accuracy* in a GRAEL society, which expresses how well the agents in the GRAEL society are agreeing on some grammatical system. *Understanding accuracy* in this sense is a measure expressing how well the GRAEL society is converging into a particular kind of probabilistic grammar.

¹Also commonly referred to as the *tuning* set.

²Differences in accuracies can nevertheless be observed, but are caused by the random resolution of ties

The measure that we are most interested in, is the one denoted as GRAEL-1 *accuracy* in Figure 5.1. It is the accuracy with which the fittest agent in the GRAEL society parses the sentences of the test set, with the fitness of an agent being a weighted average of the fitness functions described in Chapter 4 (p. 102).

Section 5.1.2 will describe a number of experimental parameters that can only be used in a setup with a validation set. In fact most experiments described in this chapter will require the use of a validation set. In such an experiment, the training set consists of 80% of the corpus, with a 10% test set, while another 10% is used as a validation set which can be used in different ways. We can calculate full society and GRAEL-1-accuracy on the validation set and use these results to control the duration of the GRAEL society (cf *infra*). The use of a validation set is also required if we want to incorporate the fitness operator we defined as *accuracy*. This measure calculates the accuracy with which an agent is able to parse a held-out set of test strings and provides an objective criterion expressing how well an agent is able to parse previously unseen data.

5.1.2 Experimental Parameters

In this section, we will take a look at the different experimental parameters that can be used as variables in the experiments with GRAEL-1: population size, i.e. the number of agents in the GRAEL society, the type of crossover operation used to share information, the method for creating new generations and the fitness functions that not only influence the development of the GRAEL society, but also determine which agent to select to parse the test set. But we will first turn to the temporal aspect and look at different methods to determine the best point at which processing in a GRAEL-society should be halted.

Defining the parameters

A matter that has so far been unaddressed is the “life-span” of a GRAEL society. Even in experiments that do not include the use of generations, a GRAEL society never finitely converges into a definite state, even though

a tendency for the agents to reach a state of convergence by agreeing on some kind of common probabilistic grammatical system can be observed. The experiments will indeed show that the parsing accuracies of the agents plateau at a certain point. But it is still not possible to determine a finite point at which the GRAEL environment should end. This however poses a problem: when exactly do we stop the language games and select the fittest agent to determine its score on the test set? In other words: how do we find out when a society is at its peak? We will propose a number of different **halting procedures** that can determine the point at which activity in the GRAEL society should be stopped and the fittest agent can be extrapolated.

The simplest method just imposes an upper limit on the **number of Language Game Runs** that are performed. Once the society has played a pre-defined number of language games, it is halted and the fittest agent is selected. Another simple method just limits the **Number of Generations**. This trivially only applies to generation-based systems. We can also determine the halting point, by looking at the agents' parsing accuracy in **relation to Training Set Accuracy**. This metric requires a validation set, because we cannot measure these values on the test set as it would violate the blind-testing principle. We can define some measure of improvement over the original training set accuracy, after which the GRAEL society is allowed to halt. This comparison can be made between (i) training set accuracy and full society accuracy or (ii) training set accuracy and the accuracy of the fittest agent on the validation set. The same kind of comparison can be made to determine the halting point in terms of the agents' parsing accuracy in **Relation to Full Society Accuracy**. This means that we define some measure of improvement of the fittest agent over the full society, with both accuracies measured on the held-out validation set.

We can also define a threshold value of agreement between agents, by looking at the average **understanding accuracy** between agents, by which we mean the average accuracy the agents obtain on each other's sentences. Once the threshold has been reached, the GRAEL society can be considered to have reached its maximum state of convergence, after which continued processing would just render more of the same. This method is closely tied to the last halting method we propose: **plateau detection**. We can detect whether or not the fittest agent's accuracy on the validation set has reached a plateau of some kind.

None of these methods however, can guarantee that the GRAEL society is halted at a global rather than at a local maximum. We will therefore propose a voting mechanism that halts the society when more than half of the halting procedures trigger a halting point. This can to a great extent counter the individual biases of the halting procedures.

Population size is another important factor in the GRAEL experiments. The number of agents in a GRAEL society will have an important effect on its outcome. A smaller society would appear to achieve a high *understanding accuracy* faster than a larger society, but a larger society will generally have the benefit of having a more diverse distribution of grammatical information. It is an important point we will make during the discussion of the experimental results that a GRAEL society does not necessarily benefit from a well converging probabilistic grammar and population size will go a long way into determining how fast and to what extent a GRAEL-1 society converges. We will experiment on population sizes of 5, 10, 20, 50 and 100 agents.

Chapter 4 discussed a number of **crossover** operations: **Language Game Crossover**, **Random Crossover** and **End-of-Life Crossover**. End-of-life crossover is closely related to the **sexual procreation** method of creating new generations (cf. infra). **Mutation crossover** in which nodes bearing different category labels are crossed over, will not be discussed, as this introduces new grammatical rules in the system and therefore constitutes an instance of mutation, which will be dealt with in Chapter 7. But we will describe some experiments that investigate the benefit of having a random crossover operation in Section 5.2.8.

Chapter 4 (p. 101) introduced the notion of **generations** in the context of the GRAEL environment. A GRAEL society can in principle develop over time without establishing different generations. In such a *Single Epoch* GRAEL society, agents are allowed to grow and expand their grammars infinitely, only limited by computational hardware constraints. But especially when experimenting on large-scale corpora, such as the WSJ corpus, one cannot allow agents to limitlessly expand their grammatical knowledge for computational reasons. We have defined two different ways of generating newborn agents out of fit agents that have reached end-of-life: **splicing** in which newborn agents are created on the basis of one agent's grammar and **crossover** which creates newborn agents using two ancestors. Apart from being highly recommendable from a computational efficiency point-of-view, we can also

consider the use of generations as a way to purge noisy grammatical information from the society. Whether new generations are created and the two different methods for doing so will be considered as variable parameters in the experiments.

We have already touched upon the `fitness functions`: efficiency, size, accuracy, understanding, understandability and internal consistency. Several experiments will be conducted that alternate weights for these different fitness functions. This not only allows us to find the optimal setting with respect to parsing accuracy, but also to investigate whether an emphasis on a particular fitness function translates into grammatical peculiarities.

5.1.3 Overview of the experiments

Obviously interaction between experimental parameters can have an effect on the development of the GRAEL-1 society and we would in principle have to exhaustively experiment on all different combinations of all values of the experimental parameters. This is however computationally intractable, since it would involve a huge amount of experiments³. We therefore make some educated guesses on the interdependence of experimental parameters to enable us to limit the number of experiments to those combinations of parameters that can reasonable be assumed to influence one another. The details of this thought exercise can be found in Appendix C.

Let us list the experimental parameters and their possible settings:

1. Corpus
 - (a) ATIS: edited version of the ATIS-II corpus. (Chapter 3, p. 53)
 - (b) WSJ: a first set of experiments was conducted on 1% sample of the WSJ-corpus. But two experiments using the standard division (Sections 02-21 as a training set, Section 22 as a validation set and Section 23 as a test set) were also conducted.
2. Population Size: 5, 10, 20, 50 or 100 agents

³A total of more than 5000 experiments.

3. Crossover Operation (Chapter 4, p. 98)
 - (a) Random Crossover Disabled
 - (b) Random Crossover Enabled: random crossover between agents possible
4. Creation of new Generations
 - (a) Single Epoch
 - (b) Sexual Generation: new agents created by end-of-life crossover between two agents
 - (c) Asexual Generation: new agents created by splicing one agent
5. Halting Procedure
 - (a) Limiting the number of language game runs
 - (b) Limiting the number of generations
 - (c) Relation of Full Society Accuracy to Training Set Accuracy (requires validation set)
 - (d) Relation of Fittest Agent Accuracy to Training Set Accuracy (requires validation set)
 - (e) Relation of Fittest Agent Accuracy to Full Society Accuracy (requires validation set)
 - (f) Understanding Accuracy
 - (g) Plateau Detection (requires validation set)
 - (h) A majority voting method using all seven aforementioned procedures
6. Fitness Function: see Table 5.1

Special attention needs to be drawn to the different possible settings for the fitness function parameter. Table 5.1 displays 10 different settings. Exhausting all combinations experimentally is again not computationally tractable, so we need to limit the number of combinations. Settings 1 to 6 isolates each different fitness function, which allows us to estimate their validity in combinations (7) to (10). Setting 7 combines the accuracy on the

	EF	SI	AC	US	UB	IC
1.	1	0	0	0	0	0
2.	0	1	0	0	0	0
3.	0	0	1	0	0	0
4.	0	0	0	1	0	0
5.	0	0	0	0	1	0
6.	0	0	0	0	0	1
7.	.2	.2	.6	0	0	0
8.	0	0	.6	.4	0	0
9.	.1	.1	0	.6	.1	.1
10.	.05	.05	.5	.3	.05	.05

EF=Efficiency, SI=Size, AC=Accuracy
 US=Understanding, UB=Understandability
 IC=Internal Consistency

Table 5.1: Settings for Fitness Function Parameter

validation set with the average of the efficiency and size fitness functions. Setting 8 combines validation set accuracy with the UNDERSTANDING fitness function that measures the accuracy of inter-agent communication. Setting 9 investigates the combination of fitness functions that do not require an extra validation set. Setting 10 combines all fitness functions.

We will illustrate the way fitness function weighting works using an example. Let us consider a 5-agent society using fitness function setting 7 (the combination of accuracy and understanding). Now each fitness function will apply a ranking to the agents in a society. Consider the extreme case in which the rankings of both fitness functions are reversed:

	US	AC
A	1	5
B	2	4
C	3	3
D	4	4
E	5	5

Now the weight attributed to the decision of each fitness function is applied

and the products are summed. The agent with the highest ranking is the fittest agent according to the combination:

	US	AC	combined
A	1 *.4	5 *.6	3.4
B	2 *.4	4 *.6	3.2
C	3 *.4	3 *.6	2.7
D	4 *.4	2 *.6	2.6
E	5 *.4	1 *.6	2.5

Ties are resolved by choosing the decision of the fitness function with the larger weight.

The **population size** parameter setting should have a significant impact on all other parameters. We therefore perform experiments for each corpus and each population size. It is suggested in Appendix C that the optimal manner in which **new generations** are created is not related to the corpus used. We can therefore determine the value of this parameter for the ATIS-corpus and use this value in our experiments with the WSJ-corpus. The **halting procedure** parameter constitutes a special case. We defined eight possible settings, but changing its value, however, does not alter the GRAEL environment destructively. There is nothing stopping us from resuming activity in the GRAEL society after halting it when one of the thresholds of one of the parameter settings has been reached. It is therefore not necessary to repeat all experiments eight times for each parameter setting.

The fitness functions are considered to be strongly correlated to the corpus parameter and the population size. Due to the long processing times on the WSJ-corpus, we chose to limit the experiments to two possible settings (settings 4 and 8 in Table 5.1). Since the other settings are more geared towards the investigation of the dynamics of grammar in an evolutionary context, rather than constitute actual workable parameter settings, this limitation should not be harmful.

The following table provides an overview of the GRAEL-1 experiments:

Corpus	Actual Experiments	Virtual Experiments
ATIS	5 population sizes * 3 generation models * 10 fitness functions = 150 experiments	* 8 halting procedures = 1200 virtual experiments
	+2 random crossover experiments * 3 generation models * 5 population sizes = 180 experiments	= 1230 virtual experiments
WSJ	5 population sizes * 2 fitness functions = 10 experiments + 2 full WSJ experiments = 12 experiments	* 8 halting procedures = 80 virtual experiments = 82 virtual experiments

The parser used in the experiments is the memory-based approximation of data-oriented parsing discussed in Chapter 3. For the ATIS corpus it was possible to employ the integrated system described on page 76. But with almost 40.000 sentences per run to parse at an estimated average parse time of 90 seconds per sentence, it was not computationally feasible to run the GRAEL-1 experiments in the same way for the WSJ-corpus. We therefore performed the 10 experiments for the WSJ-corpus on a 1% sampling of the training set, validation and test set. The optimal system was then used to perform two GRAEL-1 experiments on the entire data set.

One important addition was made to the parser: the ability to generate partial parsers during language games. Agents typically start out with a very limited and weak grammar, so that in the initial stages of a GRAEL-society it may not be possible to generate a full parse for some sentences. But since it is paramount for language games that there is at least some grammatical structure being proposed, agents are allowed to propose partial parses, such as the one in Figure 5.2 (illustrating a partial parse for the sentence *I would like a flight from Brussels to Toronto*).

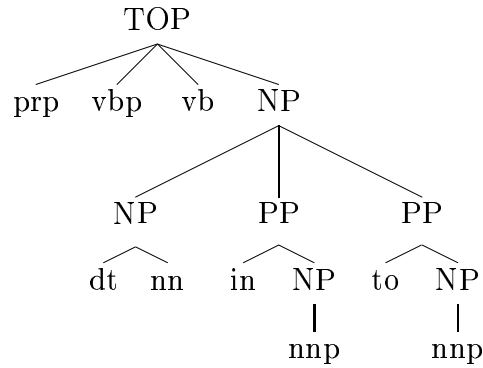


Figure 5.2: Incomplete Parse for GRAEL-1-agents

5.2 Experimental Results: the ATIS corpus

In this section, we will take a look at the experimental results of GRAEL-1 on the ATIS-corpus. Since population size is the most important experimental parameter, this section is subdivided according to population size. Experimental results on the WSJ-corpus will be discussed in Section 5.3.

For the experiments with the ATIS corpus, we used the same 10-part division as in the experiments of Chapter 3. Partition 1 (57 sentences) was used as a validation set and Partition 2 (58 sentences) was used as the test set. The remaining 463 sentences were used as a training set to be distributed over the GRAEL society. Using Partition 1 as a validation set means that we typically have a smaller training set than in the experiments of Chapter 3, which renders a direct comparison between the GRAEL-1 and the previous ATIS-experiments more difficult.

Let us first define a number of *default* settings as a convenient starting point to our discussion. This default GRAEL-society has a population size of 20 agents⁴, does not use generations⁵, only uses understanding for a fitness

⁴20 is a middle ground between the societies that have a limited amount of agents (5,10) and the societies that have a large number of agents (50,100).

⁵Fitness functions only apply to the selection of the fittest agent and not to the development of the society proper.

function⁶ (Function 4 in Table 5.1) and puts an arbitrary limit on the number of language games to determine the halting point of a GRAEL-society. In contrast to the experiments in Chapter 3, we will focus our discussion on the F-score, rather than exact match accuracy: with a 58 sentence test set, observed differences are too small on the sentence level to base our discussion on.

5.2.1 20 Agents

We will first look at the “default” population size of 20 agents in a GRAEL society and then respectively look at smaller and larger societies. We will not exhaustively go into the results of all the experiments. A full results table of all experiments on the 20-agent society can be found in Appendix D.

Single Epoch

We first consider the default experiment that implements a single epoch GRAEL environment. This means that the fitness functions play no role in the actual make-up of the society over time, since they are only used to select the fittest agent from the society and do not decide on candidates for future generations. This also allows us to bundle all experiments in which fitness functions are varied into one experimental run.

Let us first look at the fitness function of understanding (US), in which the fittest agent in the GRAEL society is the one with the highest average understanding, i.e. the best F-score observed in inter-agent communication. Figure 5.3 displays the course of the experiments: understanding accuracy starts low at around 30%, quickly climbs in the initial phase, with widely varying accuracies from one language game to the next. Over the course of about 100 runs, accuracy more or less linearly increases. At around 125 games, understanding accuracy in the corpus seems to level around 95%. This F-score can be compared to the result one would get when using the training set to parse itself⁷, since from around the 150th language game run,

⁶GRAEL revolves around language games in which this notion of understanding plays a pivotal role.

⁷This figure is displayed in Figure 5.3 as the baseline accuracy (94.9%).

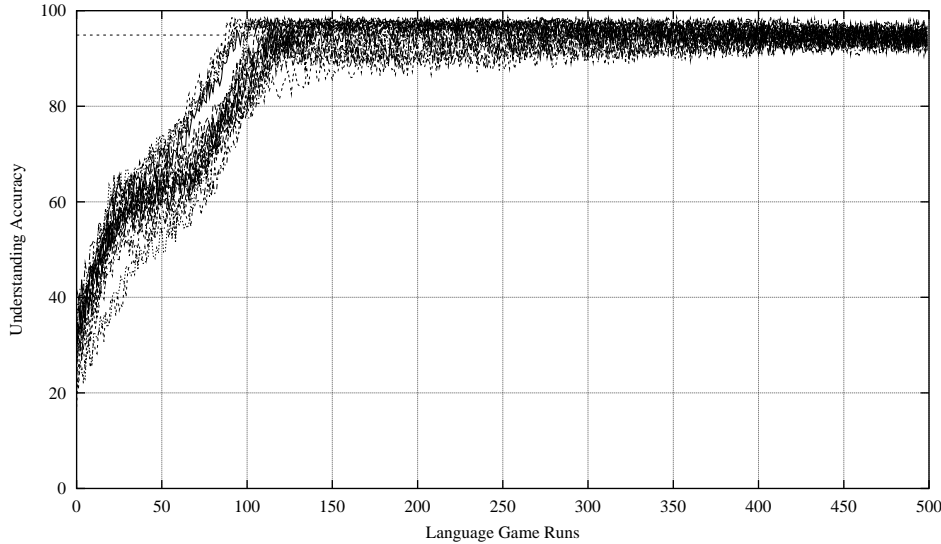


Figure 5.3: GRAEL-1 - 20 Agents - Single Epoch - Understanding Accuracy (US Fitness Function)

most of the grammatical information between the agents has been shared and there is no information in the test set (i.e. the agent to be parsed) that is not available to the training set (i.e. the agent that parses).

After about 150 runs, variance between accuracies decreases steadily over time. This indicates that the society has reached convergence and that all agents have a similar grammar, yielding similar results. The more language games are being played, the more the agents' grammars become tuned in to one another. The graph in Figure 5.3 shows that the weaker agents become stronger over time, but also that the stronger agents become weaker and are drawn towards baseline accuracy. The last 200 language game runs in Figure 5.3 show that there is hardly any development in the agents' accuracies anymore. The graph also shows that already after 150 language game runs, there is no more understanding accuracy to be gained for most of the agents, while the fittest agents in the society had already reached their peak after less than 100 language game runs. Since there are no generations, the fitness functions are only used to select the best agent to disambiguate the test set, after the society is halted.

Let us now turn to the different halting procedures. Since most of these

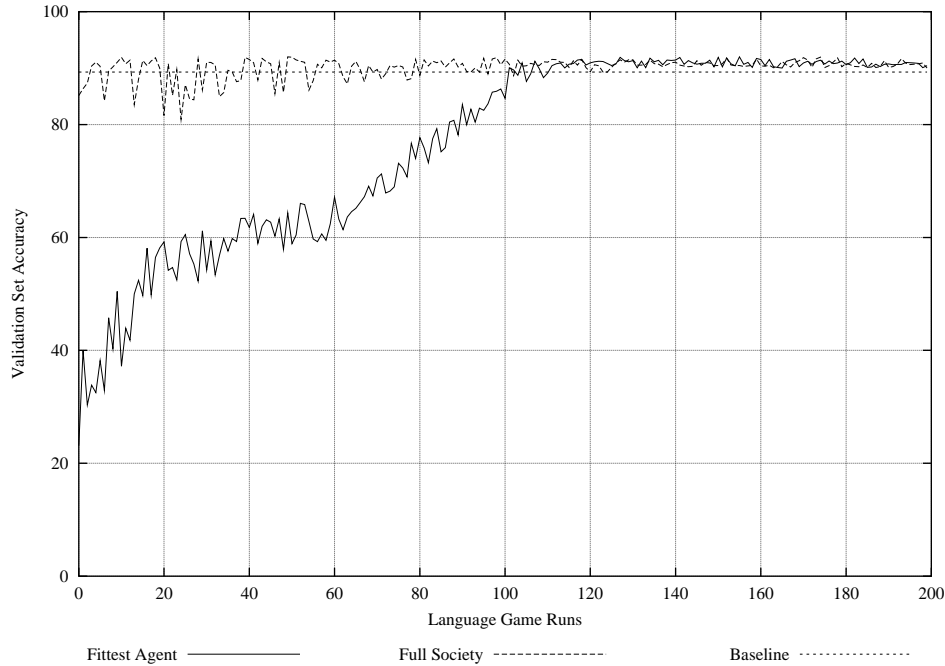


Figure 5.4: GRAEL-1 - Single Epoch - Fittest Agent Accuracy vs Full Society Accuracy vs Baseline Accuracy on Validation Set

are based on accuracy scores on the held-out validation set, we provide Figure 5.4 in which the F-score of the fittest agent on the validation set is plotted against the full society accuracy and the baseline accuracy, i.e. the F-score obtained by the initial training set on the validation set.

The default halting method we suggested earlier is to impose a limit on the number of language game runs. We halt the society after 200 language game runs for this experiment. Table 5.2 displays the results on the test set at this point. The fittest agent achieves a 97.1% F-score in its last language game with another agent in the society. He scores 91.0% on the validation set and **90.9%** on the test set. In other words, at this point a GRAEL-1 Accuracy of 90.9% (cf. Table 5.2) is achieved. This is on par with Full Society Accuracy, slightly higher than Average Agent Accuracy and significantly higher than the 89.3% baseline accuracy achieved by the original training set.

McNemar tests (Table 5.3) show that the difference between the baseline accuracy and the other accuracies is significant on the recall score, but not

200 Language Game Runs	Exact Match		Correct Constituents	LP	LR	$F_{\beta=1}$ %
	(/58)	%				
Baseline Accuracy	41	70.7	437	/483	/496	89.3
Full Society Accuracy	44	75.9	448	/491	/496	90.8
Average Agent Accuracy	43 (± 1)	74.1 (± 1.7)	444	/485	/496	90.5
GRAEL-1 Accuracy	44	75.9	447	/488	/496	90.9

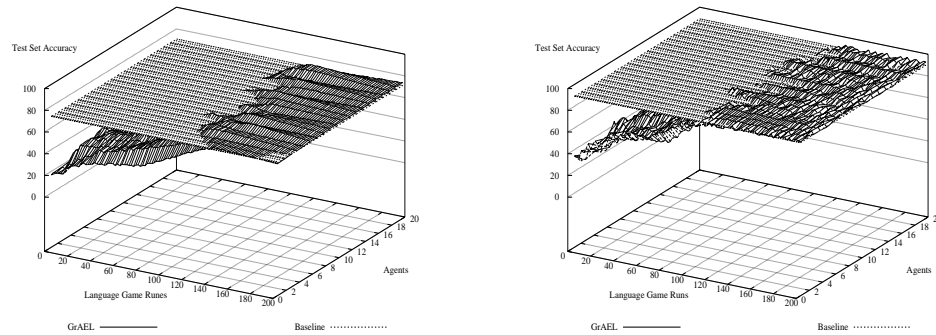
Table 5.2: GRAEL-1 - Single Epoch - US Fitness Function, after 200 LG Runs

	Exact Match		Constituents	
baseline accuracy vs GRAEL-1 accuracy	11	6	44	15
	3	38	5	432
	$P(0.2) = 0.5 > 0.05$		$P(3.9) = 0.044 < 0.05$	
full society accuracy vs GRAEL-1 accuracy	12	2	41	7
	2	42	8	440
	$P(0.3) = 0.6 > 0.05$		$P(0.06) = 1 < 0.05$	

Table 5.3: McNemar Tests for GRAEL-1 experiment

on the exact match accuracy. But note that the size of the test set does not allow reasonable statistical significance tests to be performed for exact match accuracy. The difference between full society and GRAEL-1 accuracies are not-significant. This indicates that the agents in the single epoch GRAEL-society have very similar grammars, as there is hardly a significant difference between a grammar extracted from the entire corpus and the fittest agent’s grammar. Further corroborating this is the fact that average agent accuracy has a very low standard deviation.

For this experiment, we also tested each agent's accuracy on the test set after each language game. The figures below display these accuracies compared to the baseline model. The graph shows that at around 120 language game runs, the GRAEL-society starts performing better than the baseline model.



Exact Match

$F_{\beta=1}$ -score

The halting procedure that limits the number of generations does not apply here, so we turn to the procedures that peruse the data displayed in Figure 5.4 to determine the halting point. First up is the halting procedure that measures the relation between full society accuracy and training set (baseline) accuracy on the validation set. As soon as full society accuracy on the validation set exceeds training set accuracy, the last 10 results of the full society are recorded. Unless training set accuracy exceeds full society accuracy again, the society is halted if the standard deviation on the last 10 F-scores is lower than 1.0%, suggesting that there is not much more room for improvement. This measure would halt the society at an early point in the current experiment, namely after the 90th run. The fittest agent when the GRAEL society is halted at this point, achieves an exact match score of **69.0%** and an F-score of **83.4%**, which is significantly lower than the result obtained by the default halting procedure.

The halting procedure that looks at the relation between the fittest agent and the training set accuracy operates in a similar fashion, using the accuracy of the fittest agent on the validation set instead of full society accuracy. This procedure halts the society at a later point in time (run 120). The fittest

agent of the GRAEL-society at this point achieves an F-score of **91.1%** on the test set, which is higher than the accuracy of the fittest agent at run 200, even though this can not be statistically corroborated by the McNemar test.

The relationship between fittest agent accuracy and full society accuracy on the validation set is problematic in this experiment, as even after fittest agent accuracy starts exceeding full society accuracy, the two plots are still interspersed (cf. Figure 5.4) over time, with full society still outperforming the fittest agent at random intervals. A halting point does occur finally at run 143, with the fittest agent achieving a **91.0%** F-score on the test set.

The plateau detection procedure tries to limit the amount of parsing on the validation set, by not looking at the relationship between the accuracy of the fittest agent with some other kind of accuracy. It simply tries to pinpoint some kind of plateau of accuracy of the fittest agent on the validation set. Despite limiting the amount of computation involved, this procedure is vulnerable to halting the society at a local maximum, since there is no indication of a threshold value for the performance of the fittest agent as is the case for the previously described halting procedures. In this experiment however, plateau detection does not need to deal with this problem: it halts the GRAEL-society at run 114, obtaining a **91.1%** score.

Another method which further reduces the amount of computation by totally eliminating the need for a validation set, looks at the understanding accuracies recorded during the language games. For each run of language games, we record the average understanding accuracy, i.e. the average F-score during the inter-agent communication. If the average understanding accuracy does not exceed the mean value of the last 10 average understanding accuracy scores \pm the standard deviation, convergence is assumed and the society is halted. In this experiment, this point occurs after about 146 runs (also see Figure 5.3 for an indication). But again, differences in parsing accuracy are not statistically significant, with an exact match accuracy of **75.9%** and an F-score of **91.0%**.

The final halting procedure uses majority voting. The GRAEL-society is halted when 4 out of 7 procedures have triggered a halting point. Table 5.4 displays the different halting points and the respective accuracies the fittest agent obtains on the test set. The halting point according to the majority voting procedure occurs at language game run 143.

Halting Procedure	Halting Point	Exact Match	F-score on Test Set (%)
Relation FS Accuracy vs TS Accuracy	90	69.0	83.4
Plateau Detection	114	77.6	91.1
Relation FA Accuracy vs TS Accuracy	120	77.6	91.1
Relation FA Accuracy vs FS Accuracy	143	77.6	91.0
Understanding Accuracy	146	77.6	91.0
Limit number of language game runs	200	75.9	90.9
Limit number of language game runs	500	75.9	90.9
Limit number of generations	—	—	—

Table 5.4: GRAEL-1: Single Epoch: Halting Points for understanding Fitness Function

Even though the increase in accuracy on the test set is not statistically significant when compared to the default halting procedure that imposes a limit on the number of language game runs, the number of games that need to be played can be reduced from 200 to 143, without any loss in accuracy. In fact, accuracy is higher at the majority voting halting point. This is typical for a GRAEL-1 experiment as we will see: there is a small period of time, right before the society reaches a state of convergence, in which the agents' performance is at its peak. We will come back to this effect in Section 5.2.6.

With the description of this default experiment, we have mainly tried to introduce the practical aspects of some of the parameters we discussed in Section 5.1.3. It is interesting to note that with this very simple approach, we are already able to improve on the original grammar with a fairly basic system and a limited amount of processing time. The improvement can be attributed to the practical context that GRAEL provides: the redistribution of the probability mass occurs on the basis of observations made during **parsing**. So rather than mirroring the distribution of the original data set, the probabilistic values of the grammar rules now reflect useful values for the task of parsing itself.

Let us now turn to the different fitness functions and look at their effect on the course and accuracy of a GRAEL society. Table 5.5 displays the results of all the experiments on the single epoch GRAEL-1-environment. The fitness function of size measures the agent's grammar and singles out the agent with the most compact grammar as the fittest agent. Not surprisingly, this fitness function does not produce well-performing agents. Even in the convergence state after 200 runs, the fittest, i.e. most compact agent roughly achieves

baseline performance, indicating that the more grammatical structures an agent possess, the higher it will score on the test set. The data-analysis in Section 5.2.9 will provide an insight as to whether or not size matters vis-a-vis performance.

Even though it would intuitively appear to be closely related to size, the efficiency fitness function, which measures the average CPU-time each agent uses for parsing, achieves a significantly better result⁸. Even though it is outperformed by most other fitness functions at the majority voting halting point, it does achieve remarkably good scores very early in the lifespan of the GRAEL-society.

Understandability, which expresses the agent's fitness in terms of how well other agents parse its E-language, does not constitute a good fitness function if we want a well performing grammar. This is hardly surprising, as the make-up of the E-language only initially influences the I-language that is being developed. We will see in subsequent experiments that understandability functions, more as a random choice in the society, rather than as a guided principle.

The accuracy fitness function, which assumes the fittest agent to be the one with the highest score on the validation set, would intuitively appear to yield agents that score equally well on the test set, even though there is a distinct danger of over-fitting. The data nevertheless shows that this fitness function does perform well: the majority voting halting procedure produces an agent that achieves a **91.1%** F-score.

Internal consistency is not a very good fitness function, as it does not make allowance for the parsing of any kind of unseen data. The aim of implementing this fitness function was to see how far one can get by defining the fitness of an agent strictly on its own terms. This particular experiment, with its high degree of convergence, is however not well suited to draw any conclusions on this issue, as even the worst fitness function will still select an agent that performs reasonably well.

Looking at the data in Table 5.5, it would seem that little is to be gained when fitness functions are combined in a single-epoch society. But we would

⁸Note that the recorded CPU-time may also be influenced by uncontrollable factors that are unrelated to GRAEL-1.

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	90	80.1	—	—	—	—	115	89.9	146	89.9	200	90.2	—	—	200	90.2
Efficiency	90	88.2	106	90.8	—	—	104	90.7	146	90.9	200	90.9	—	—	146	90.9
Accuracy	90	84.2	131	91.1	131	91.1	131	91.1	146	91.0	200	91.0	—	—	131	91.1
Understanding	90	83.4	120	91.1	143	91.0	114	91.1	146	91.0	200	90.9	—	—	143	91.0
Understandability	90	78.6	138	90.3	—	—	120	87.3	146	90.7	200	90.8	—	—	146	90.7
Internal Consistency	90	72.3	—	—	—	—	149	90.6	146	90.7	200	90.8	—	—	200	90.8
Efficiency & Size + Accuracy	90	75.4	143	91.1	143	91.1	143	91.1	146	91.1	200	91.1	—	—	143	91.1
US&UB + Accuracy	90	89.9	98	90.7	107	91.1	98	90.7	146	91.0	200	91.1	—	—	107	91.1
Efficiency & Size + US&UB	90	82.4	131	91.1	137	91.1	130	91.0	146	91.1	200	91.1	—	—	137	91.1
Efficiency & Size + Accuracy + US&UB	90	80.4	136	91.1	137	91.1	136	91.1	146	91.0	200	91.1	—	—	137	91.1

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table 5.5: GRAEL-1 ATIS-20 Agents - Single Epoch - Halting Points and F-scores on Test Set

like to draw attention to the experiment that combines the efficiency, size, understanding and understandability fitness functions. The majority voting method to halt the society, produces a fittest agent that achieves a **91.1%** F-score on the test set. This combination seems able to yield a strong agent for parsing, without having to process a validation set, which provides an advantage from a computational point of view. And when combining understanding and accuracy, the system is able to reduce the number of language game runs by almost 50% with no loss of accuracy.

One last point with respect to Table 5.5 is that the majority voting method is generally able to halt the GRAEL-society at its peak performance. No single halting method seems to consequently suggest the best halting point, but the combination of different sensibilities of what condition the GRAEL-society should be in to be halted, seems to be a valid method.

We will now turn to the experiments that introduce generations in the GRAEL-society. We have suggested two method of creating new generations: **asexual procreation** (splicing), in which fit agents are allowed to splice into two agents, and **sexual procreation**(crossover) in which two fit agents cross over their grammars to create three new agents. Both methods have in common that a number of unfit agents will need to disappear from the society, taking their suboptimal grammatical information with them. This has as a direct effect, that as opposed to the previous experiments, fitness functions in fact do shape the development of a GRAEL-society. This requires each fitness function to have his own experimental run.

But first we need to resolve a fundamental issue: both methods for creating new generations need a way to establish the end of an agent's life, i.e. the point at which it is appropriate to procreate. This can also be considered as a parameter to be optimized and future research will look into alternative approaches. But for the experiments described in this chapter, lifespan is defined in terms of the number of new rules an agent acquires during language games. After each language game run, the number of new grammatical rules that each agent has learned is recorded. The lower this number, the "older" an agent is considered to be. If over the course of n language game runs, an agent does not obtain any new grammatical information from other agents, it is assumed that it has all the information that is available in the society and is therefore considered to be an end-of-life agent. n is usually set to the number of agents in the society: this provides the agent with a buffer period

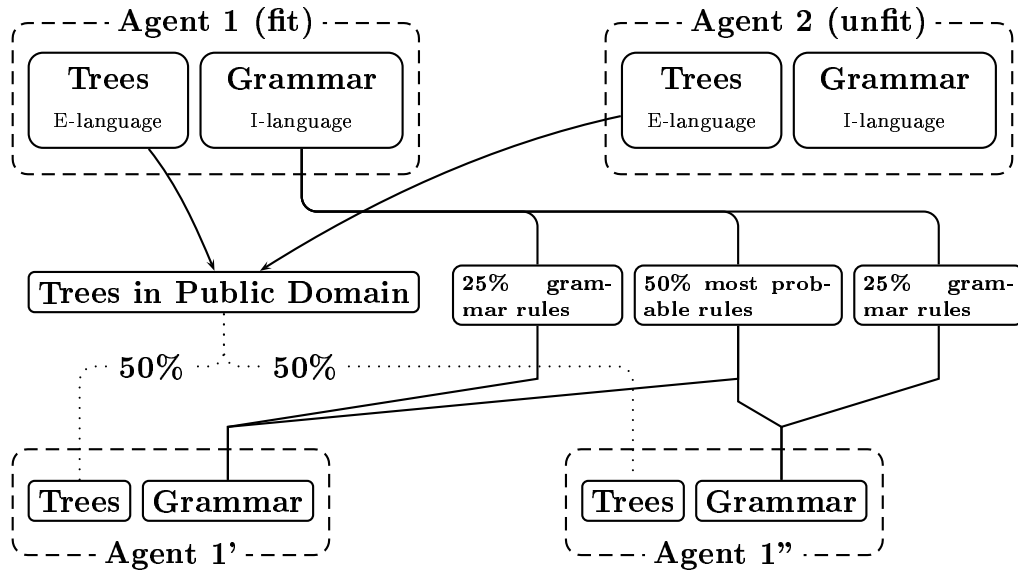


Figure 5.5: Splicing an Agent

during which it can engage in language games with most of the other agents in the society to further optimize the grammatical information it contains before reaching its end-of-life state. Next, the end-of-life agent is allowed to procreate, either by splicing itself or by crossing over with another agent.

Asexual Procreation (Splicing)

Let us first look at the experiments on asexual procreation (described in Figure 5.5): after each language game run, the end-of-life agents are rounded up and evaluated: if they belong to the 50% fittest agents in the society, they are eligible for procreation. For each agent to be spliced, the oldest agent among the 50% most unfit agents will disappear from the society. Before it does, however, both the fit agent as well as the unfit agent, let go off their E-language, i.e. their trees to be parsed by other agents, sending them to what is described in Figure 5.5 as the *public domain*. The I-language, i.e. all the grammatical structures that **agent1** has compiled by playing language games with other agents is divided into three parts: 50% of the grammar, containing the most probable rules, and two 25% parts in which

the remaining 50% are randomly distributed. Each agent receives the 50% part and one of the two 25% parts. Now there are two new agents, **agent1'** and **agent1''** each containing 75% of their ancestor's grammar.

The newborn agents **agent1'** and **agent1''** now each have their own I-language, but still need an E-language. They can obtain this by randomly pulling trees from the public domain. To make sure all the sentences of the E-language are accounted for by the I-language, the grammatical structures are extracted from those trees and added to the I-language.

In the experiments described here, the use of generations does not dynamically alter the population size: there is a fixed number of slots for agents. The asexual procreation method however creates two new agents. One of the agents can take the ancestor's slot in the society, while the other agent will take the place of the oldest unfit agent in the society. The latter will disappear from the society. We put an arbitrary limit of 10 generations per slot in the society. This means that the society is halted when there is only one agent left in the society and consequently no more language games can be played.

Caveat It should be noted that at this point in the experiments, the method for creating new generations is in no way trying to mirror biological processes. Since the goal of GRAEL-1 is to optimize the probability mass of grammars induced from annotated corpora, so that they can be used to optimally parse new, unseen data, the way in which old agents procreate, disappear and originate is motivated by practical purposes, rather than theoretical.

- The strict distinction and separation between the E-language and the I-language does not agree with psycholinguistic insights on human language analysis and/or generation. But allowing the I-language to influence the E-language, would necessarily introduce erroneous grammatical structures in the agent's E-language, rendering them unsuitable as a touchstone in language games with other agents.
- The E-language in unfit agents does live on in new generations, which violates the concept of a generation-based system. If we would however remove the E-language as well as the I-language when creating new agents, the grammatical information that is present in the GRAEL-society at later stages, would become very limited, as agents would prefer a small set of easily parsable structures. This would improve convergence and understanding scores throughout the corpus, but would yield unsuitable grammars for parsing unseen data
- Newborn agents obtain 75% of their ancestor's grammar. This figure is not in the slightest motivated by biological evidence, but is a reasonable ballpark figure that helps the society move along at a reasonable pace, while leaving room for improvement over the original grammar
- Newborn agents compile their E-language from the so-called "public domain", which consists of fit, as well as unfit agents' structures. This method ensures that grammatical information, that may previously have yielded unfit agents is redistributed across the society, so that other agents may still benefit from these structures.

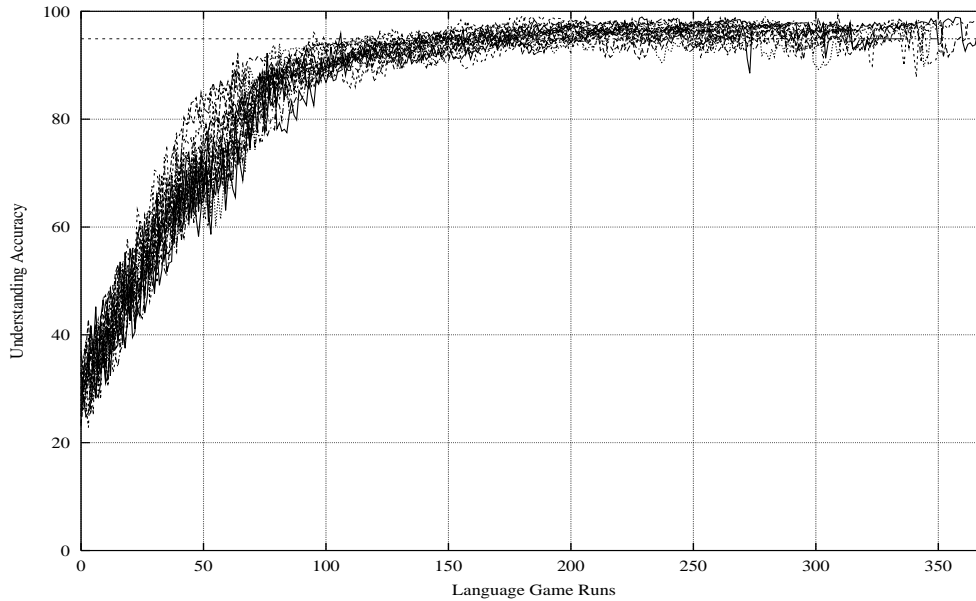


Figure 5.6: GRAEL-1 - 20 Agents - Splicing - Understanding Accuracy (US Fitness Function)

First, we look at the default experiment which uses understanding as the only operative fitness function. Figure 5.6 plots the understanding accuracies of the agents. The baseline accuracy is the accuracy of using the training set to parse itself. The plots start out by running similarly to those in Figure 5.3, but at around the 50th language game run, the first agents start splicing. The overall effect is that the peak of the society in understanding accuracy is delayed to around the 150th language game run as opposed to the 100th language game run in the previous experiments.

After about 200 language game runs, understanding accuracy levels out and the society seems to have reached a state of convergence. About 120 language game runs later, most agent slots have produced their 10 generations and the number of agents starts to decrease. This has as an effect that for a brief period, until the last agent disappears, understanding accuracy rises again, as the small number of agents seem to have an easier time adjusting to each other's trees. The rise in understanding accuracy does not mean that these agents necessarily constitute better parsers however.

200 Language Game Runs	Exact Match		Correct Constituents	LP	LR	$F_{\beta=1}$ %
	(/58)	%				
Baseline Accuracy	41	70.7	437	/483	/496	89.3
Full Society Accuracy	43	74.1	445	/482	/496	91.0
Average Agent Accuracy	43 (± 0.9)	74.1 (± 1.5)	447	/485	/496	91.1
GRAEL-1 Accuracy	45	77.6	455	/495	/496	91.8

Table 5.6: GRAEL-1 - Splicing - understanding Fitness Function, after 250 LG Runs

Table 5.6 provides an overview of the results on the test set after the 250th run. Baseline Accuracy is the same as in the previous experiment. Full Society Accuracy and Average Agent Accuracy are higher than the baseline accuracy. The accuracy of the fittest agent at this point however is considerably higher than all other accuracies, with a considerable improvement in F-score over the previous experiment as well. This is mainly due to a significantly increased precision score. As in the previous experiment, McNemar significance tests on these figures, show that the difference between baseline accuracy and GRAEL-1-accuracy is significant for the F-score, while this is not the case for the difference between the full society and the fittest agent.

Figure 5.7 shows the accuracy of the fittest agent on the validation set, plotted against the accuracy of the full society and the baseline accuracy. Except for a slight lag around the 50th language game run (the point at which many agents produce their first offspring), we notice that the transition between generations is not clearly visible in Figure 5.7, because it is not one particular agent’s plot being displayed, but the plot of the fittest agent after each language game run. Compared to Figure 5.4, we notice that the fittest agent accuracy takes about 50 runs longer to catch up to the baseline model and the full society, but then overtakes it more convincingly than the fittest agent in the previous experiment.

The different halting points for this experiment and the accuracy of the fittest agent at that point on the test set can be found in Table 5.7. In the current experiment, in which understanding is the only fitness function at work, the majority voting method would halt the society after the 186th run with an F-score of **91.7%**. This is only slightly lower than the score that is obtained when halting the society after 250 runs. We have already noted that at the end of the society, when only few agents are left, there is a danger of overfitting, with only limited grammatical resources available in

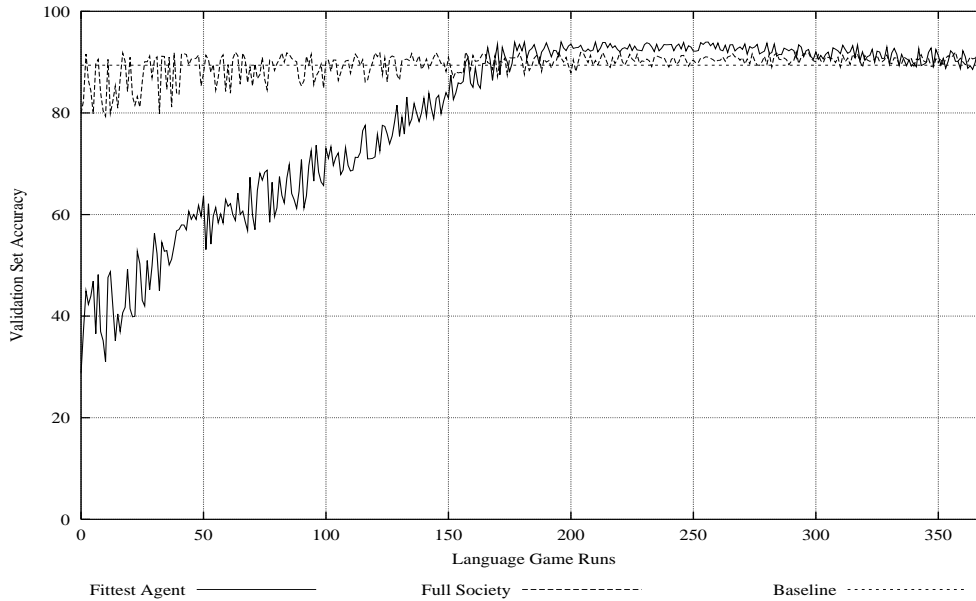


Figure 5.7: GRAEL-1 - Splicing - Fittest Agent Accuracy vs Full Society Accuracy vs Baseline Accuracy on Validation Set

the society. The agents at this point have been tuned to one another, but do not hold enough grammatical structures to achieve the accuracy of their ancestors. It is apparent that for this kind of experiment, the point at which the society is halted is paramount to finding an agent with a good grammar for parsing unseen data.

Let us take a look at the other fitness functions. Table 5.7 shows that the size fitness function does not produce a good GRAEL-society. Agents that have a lot of grammatical knowledge are not allowed to splice, so that a typical newborn agent holds less grammatical knowledge than newborn agents in other experiments. As a consequence, this experiment takes significantly longer than most other experiments, as new generations are created in longer intervals. Since only end-of-life agents, containing a fair amount of grammatical information, are selected for procreation, it would seem that the effect of the fitness function on the society as a whole would be limited. But the fitness function is also responsible for selecting the fittest agent to parse the test set and since at some points in the society no end-of-life agents are present, this selection is not limited to end-of-life agents, so that overall

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	—	—	203	76.9	250	80.3	381	78.5	381	78.5
Size (10%)	—	—	—	—	—	—	—	—	182	83.6	250	86.6	381	82.3	381	82.3
Efficiency	—	—	—	—	—	—	—	—	188	86.3	250	89.1	353	85.9	353	85.9
Accuracy	180	91.8	235	92.0	235	92.0	81	63.3	181	91.8	250	91.9	341	91.7	235	92.0
Understanding	209	91.8	186	91.7	186	91.7	186	91.7	176	91.5	250	91.8	370	91.1	186	91.7
Understandability	—	—	247	91.2	257	91.2	177	89.1	210	90.4	250	91.2	336	89.9	250	91.2
Internal Consistency	153	78.3	233	90.6	235	90.6	233	90.6	201	89.9	250	90.5	351	89.5	233	90.6
Efficiency & Size + Accuracy	187	90.2	229	91.6	229	91.6	229	91.6	187	90.2	250	91.4	362	90.7	229	91.6
US&UB + Accuracy	188	90.5	237	92.1	237	92.1	237	92.1	193	90.6	250	92.0	353	90.4	237	92.1
Efficiency & Size + US&UB	241	91.6	246	91.7	—	—	60	58.9	205	91.2	250	91.6	379	90.0	246	91.7
Efficiency & Size + Accuracy + US&UB	209	92.0	198	91.2	268	91.7	198	91.2	211	92.0	250	91.8	361	91.0	211	92.0

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table 5.7: GRAEL-1 ATIS - 20 Agents - Splicing - Halting Points and F-scores on Test Set

newborn agents are selected for parsing.

When we limit the number of agents eligible for selection as the agent to parse the test set to the 10% oldest agents in the society, F-scores do increase at some points in the society. But there are still not enough halting points to trigger majority voting at any other point than the end of the society, at which point there are only two agents left, so that accuracy suffers anyhow. The results of this experiment suggest that the fitness function of size for selecting agents eligible for procreation, does in fact have a strong effect on the outcome of a society.

Experiments described later in this chapter also corroborate this claim: in these experiments, the selection of agents for procreation and the selection of agents for parsing are handled by different fitness functions. These experiments show that even when a strong fitness function like `accuracy` is allowed to pick the agent for parsing the test set, the fitness function of size for choosing candidates for procreation has weakened to society to such an extent that there are no agents in the society that can parse the test set well.

The `efficiency` fitness function fares a little better in the splicing experiment, but still does not achieve anything close to baseline accuracy, as opposed to its single epoch counterpart. In the latter experiment a high degree of convergence was apparent, with most agents having a similar grammar early on, so that the `efficiency` fitness function was an expression of real efficiency issues in the grammar, rather than be correlated to grammar size. In this experiment however, agents throughout the society widely vary in grammar size, so that the `efficiency` function is more related to grammar size, than to its internal structure. To its credit however, `efficiency` does outperform the fitness function of size by a significant margin, proving again that the size of a grammar is only partially related to the amount of CPU-time parsing with that grammar entails.

Using `understandability` as a fitness function achieves results that are better than the baseline model, but are still lower than other fitness functions. The `understandability` measure expresses how well other agents' I-languages (to be optimized) are suited to parse the agent's E-language (constant). Since newborn agents obtain their E-language from a public domain unit in which trees not used by the society reside, the `understandability` fitness function prefers agents that have a E-language holding a collection of easy to parse trees.

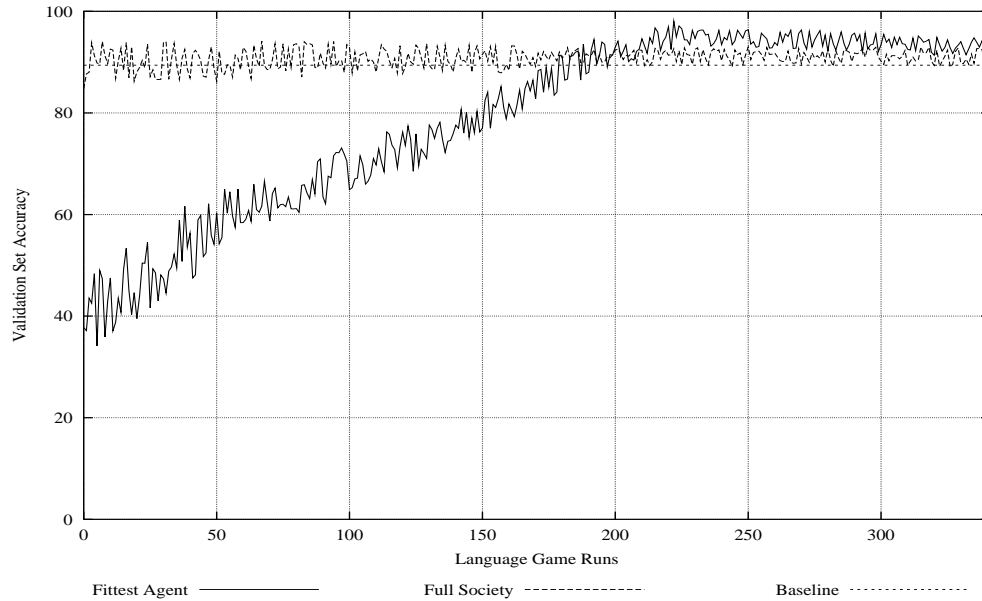


Figure 5.8: GRAEL-1 - Splicing - Fittest Agent Accuracy vs Full Society Accuracy vs Baseline Accuracy on Validation Set: accuracy fitness function

This however in no way guarantees that these agents themselves have decent I-languages. On the contrary: these agents seem to perform significantly worse than the agents that yield a good understanding accuracy.

The fitness function of internal consistency produces similar results to its understandability counterpart. We hypothesize that understandability and internal consistency fitness functions produce fittest agents that are similar to average agents in a single-epoch society with understanding as a fitness function.

Figure 5.8 shows the plot for the fittest agent in a GRAEL-society using the accuracy fitness function. The fittest agent plot linearly increases to overtake the baseline accuracy, as well as the full society around the 200th language game run. Even though accuracy decreases over time, the F-score of the fittest agent at the end of the society (341 runs) is still pretty high. Halted about 100 runs earlier, as proposed by the majority voting method, the F-score is **92.0%** which constitutes a significant increase over the baseline accuracy.

Combining the efficiency and size fitness functions with understanding and understandability⁹ does not deteriorate nor improve performance. The accuracy fitness function does not benefit from a combination with efficiency and size, but the combination with understanding does give it a slight edge over the other experiments. The combination of all fitness functions finally achieves a F-score of **92.0%**.

Sexual Procreation (Crossover)

The last set of experiments for the 20-agent GRAEL-society involves sexual procreation, the method in which new generations of agents are created by crossing over the grammatical information found in two fit end-of-life agents. Figure 5.9 describes how this works. This procedure requires two end-of-life agents. If only one is available, it will continue to live on in the society until another fit end-of-life agent becomes available. The sexual procreation method used in these experiments creates three new agents. This means that besides the two end-of-life agents, an unfit agent will need to disappear from the society. Note the difference with the splicing method in which for each fit agent and unfit agent also has to disappear from the society.

The two fit end-of-life agents and the unfit agent donate their E-language to the public domain, so that the three newborn agents can randomly draw tree-structures from it to compile a E-language of their own. The two fit agents divide their grammars into two parts: **part1** contains the 25% most probable rules in the grammar¹⁰ and **part2** holds the rest of the grammar.

part2 is further randomly subdivided into three divisions. Each of the three newborn agents will contain both ancestors' **part1** and a subdivision of each ancestor's **part2**. This means that each newborn agent gets roughly two 25% parts of the grammar from each ancestor. This does not mean however that the newborn agent's grammar is the same size as the ancestor's, since a lot of the rules in the inherited grammar parts will overlap. On average, the newborn agents' I-languages are 75% to 85% the size of their ancestors.

The default experiment using *understanding* as the only fitness function (Figure 5.10) runs a similar course as its splicing counterpart: the best un-

⁹The fitness functions that do not require the use of a validation set.

¹⁰In the case of PMPG-type rules: their subordinate clauses as well.

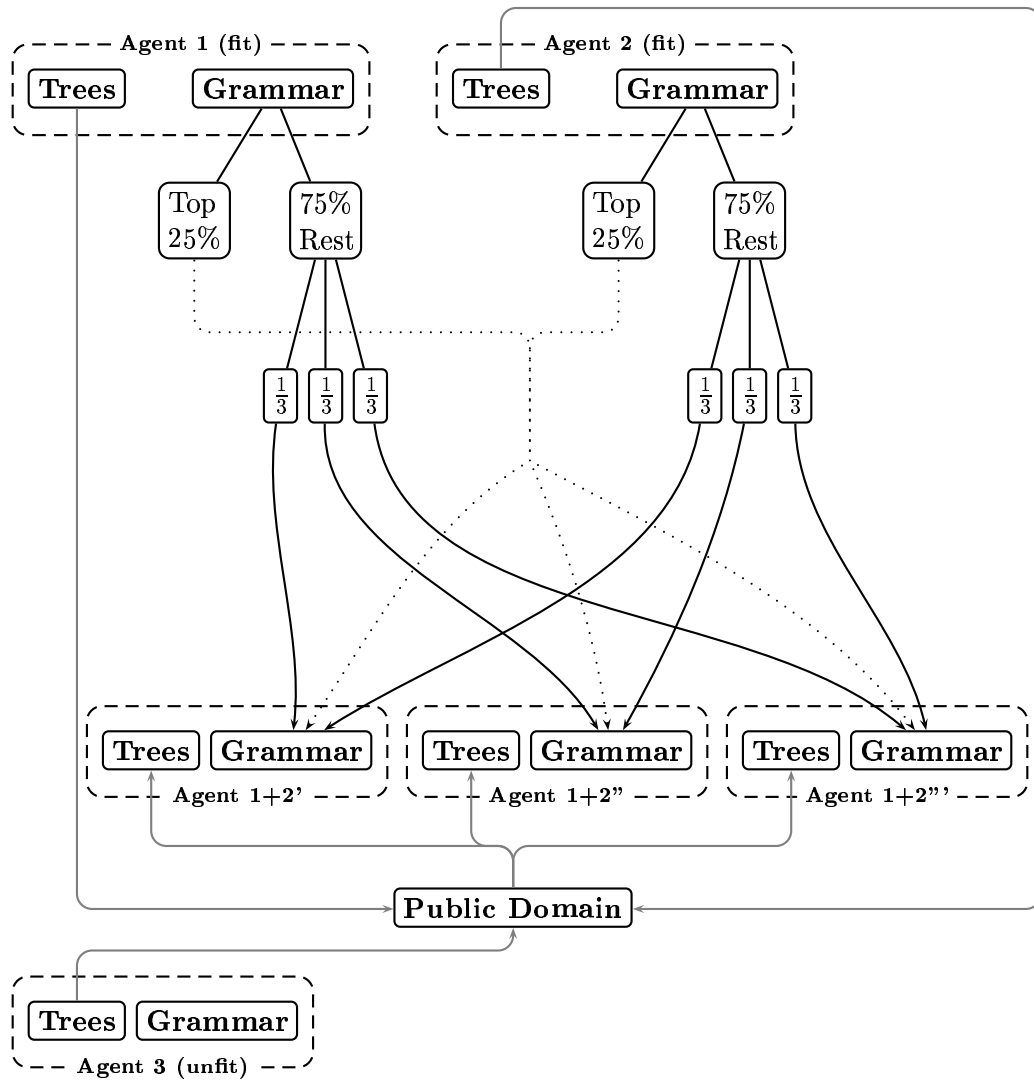


Figure 5.9: 2 Agents Crossing Over grammatical Knowledge (Sexual Procreation)

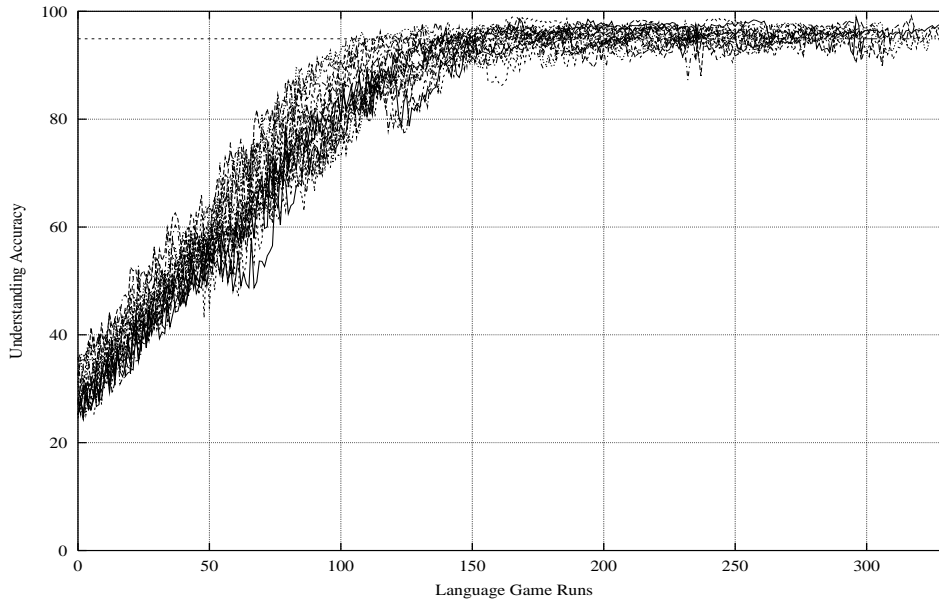


Figure 5.10: GRAEL-1 - 20 Agents - Crossover - Understanding Accuracy (understanding Fitness Function)

derstanding accuracy scores are about the same. The agents in the splicing experiment seemed to converge around the 100th language game run, while in this experiment it takes about 50 runs longer. Differences in understanding accuracies before convergences vary more than in the splicing experiment. This overall experiment however lasts about 50 runs less, which is due to the fact that newborn agents from sexual procreation hold more grammatical information on average than spliced agents.

Using majority voting to determine the halting point for this society, the fittest agent in this society achieves a **91.5%** F-score on the test set, which constitutes a 0.2% improvement over the same society in the splicing experiments. The size fitness function performs much better in these experiments: 85.3% as opposed to 78.5%. The efficiency fitness function remains unaffected by the different generation-method and so is the accuracy fitness function, even though in these experiments the GRAEL-society needs over 30 runs less to achieve the same results. The results of combining fitness functions are on par with the previous experiments. The combination of efficiency, size and accuracy benefits from the fact that the size fitness function

Method	Runs	Exact Match		Precision	Recall	$F_{\beta=1}$ %
		/58	%			
Baseline	1	41	70.7	437/483	437/496	89.3
Single Epoch	107	44	75.9	450/492	450/496	91.1
Splicing	237	45	77.6	459/501	459/496	92.1
Crossover	181	45	77.6	456/494	456/496	92.1

Baseline	Single Epoch
41	18
5	432
$P(6.3) = 0.01$	

Baseline	Splicing
26	33
11	426
$P(10.0) = 0.002$	

Baseline	Crossover
32	27
8	429
$P(9.3) = 0.002$	

Single Epoch	Splicing
36	10
1	449
$P(5.8) = 0.02$	

Single Epoch	Crossover
38	8
2	448
$P(2.5) = 0.11$	

Splicing	Crossover
32	8
5	451
$P(0.3) = 0.58$	

Table 5.8: GRAEL-1 ATIS - 20 Agents

seems to be more suited to this type of procreation. The combination of understanding and accuracy achieves a **92.1%** F-score (similar to splicing), but had the society halted one run later, 0.1% could have been gained still (even though the difference is insignificant). The F-score of the two other combinations are at least 0.1% lower than in the splicing experiment. This may be due to the fact that the majority voting method halted the society a little too early in both experiments.

Summary

This concludes the experiments for the GRAEL-society consisting of 20 agents. Table 5.8 summarizes the results. The GRAEL-1 results are based on the societies using majority voting to determine the halting point and the combination of understanding and accuracy as a fitness function. Table 5.8 also shows McNemar tests on the constituent level (i.e. recall). These show that all differences in results between the baseline model and the GRAEL-1 instantiations are significant. The difference between the single epoch method and the splicing method is significant, but the difference with the crossover method can not be substantiated on the basis of F-score alone.

To summarize, we can state that the 20 agent GRAEL-1 society is a good

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
10 Agents	90.3	90.8	91.1	91.3	90.3	90.6	91.2	91.3	91.2	91.1
20 Agents	89.9	90.9	91.1	91.0	90.7	90.8	91.1	91.1	91.1	91.1

Table 5.9: GRAEL-1 - Single Epoch: Majority Voting Scores for different fitness functions

method for grammar optimization. Using a simple method that does not include generations, we are already able to significantly improve the F-score over the baseline model. A further significant improvement is achieved by introducing a way for the society to create new generations, so that badly compiled collections of grammatical structures can disappear from the society to make way for new agents containing the start of an already well-tuned grammar. We have discussed two ways of creating new generations through asexual (splicing) and sexual (crossover) procreation. Though there is little to choose between the two of them in terms of accuracy, the crossover method reduces processing times considerably without a significant loss in accuracy.

5.2.2 10 agents

We now turn to the next set of experiments, in which we lower the number of agents in the GRAEL-society. We will not go into great detail, but concentrate on the similarities and differences between the 10-agent society and the 20 agent society. The full results table for all the experiments with the 10-agent GRAEL-1-society can be found in Appendix D.

Single Epoch

Figure 5.11 shows the course of the default experiment, with a Single Epoch 10-agent GRAEL-1 society using *understanding* as the fitness function. Since this society has fewer agents, each agent holds a larger E-language, so that grammatical information is typically distributed faster throughout the society, now that agents parse more sentence per language game. As a consequence, this society reaches convergences much faster: after 75 runs, all agents' accuracies circle around the baseline mark.

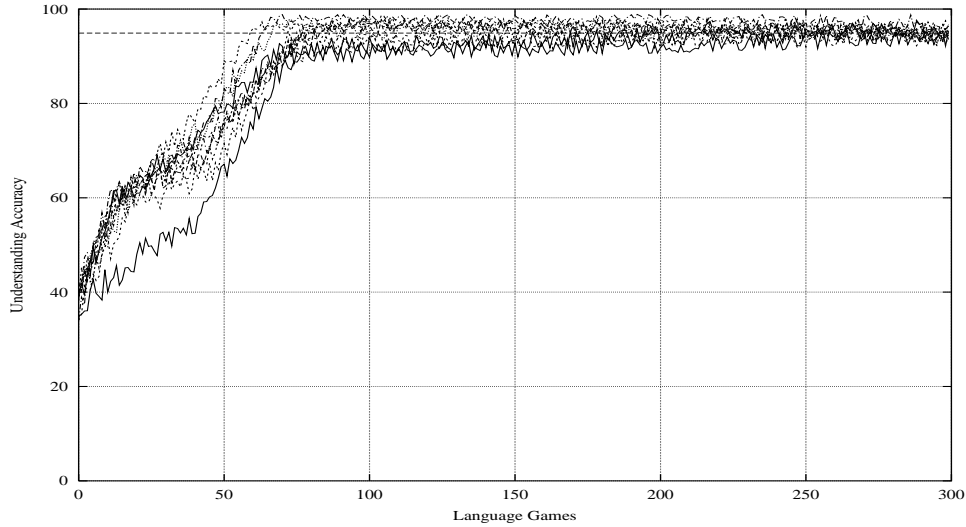


Figure 5.11: GRAEL-1 - 10 agents - Single Epoch - Understanding Accuracy (understanding Fitness Function)

Table 5.9 displays the F-score on the test set for different fitness functions and the societies discussed so far¹¹. The same tendencies are apparent: the size and efficiency functions seem to choose less than optimal agents for parsing, even though this difference evens out at the end of the society, when the agents' grammars have converged and there is little left to choose among them. The accuracy and understanding fitness function and the combinations with other fitness functions all perform reasonably well. Overall, the difference with the 20-agent society is marginal, but the results do appear to be consistently better. As the 10-agent society seems to converge faster than the 20-agent society, this would make the 10-agent society more suited as a single epoch GRAEL-society.

Asexual Procreation (Splicing)

The splicing experiment for the 10-agent society brings to light a couple of pertinent differences with the 20-agent society: not surprisingly the society

¹¹Abbreviations used in Table 5.9: **SI**: Size, **EF**: Efficiency, **AC**: Accuracy, **US**: Understanding, **UB**: Understandability, **IC**: Internal Consistency, **C1**: EF/SI + AC, **C2**: US + AC, **C3**: EF/SI/UB + US, **C4**: EF/SI/UB/IC + US + AC.

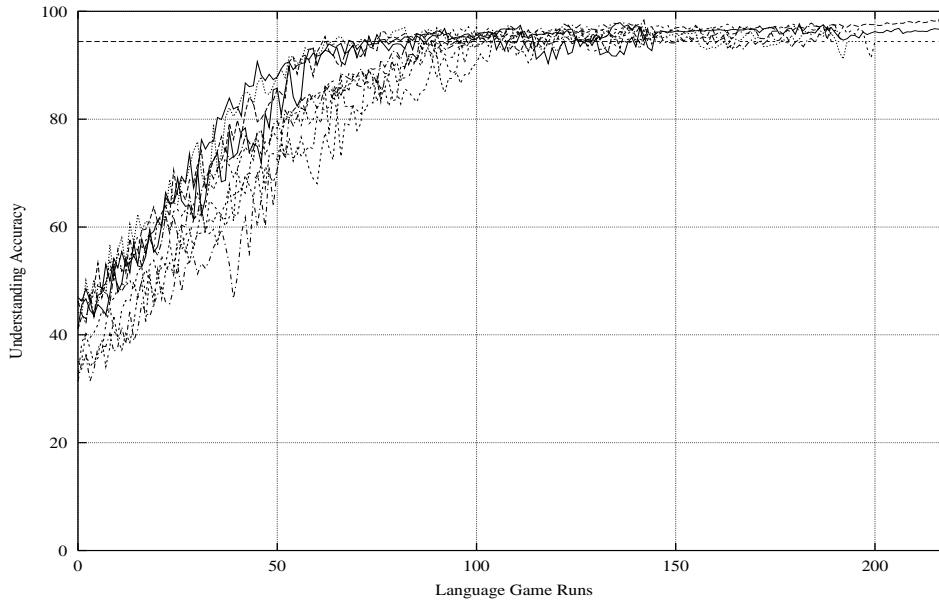


Figure 5.12: GRAEL-1 - 10 agents - Splicing - Understanding Accuracy (understanding Fitness Function)

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
10 Agents	82.9	82.9	91.9	91.9	89.6	90.7	91.6	92.0	91.8	91.7
20 Agents	78.5	85.9	92.0	91.7	91.2	90.6	91.6	92.1	91.7	92.0

Table 5.10: GRAEL-1 - Splicing: Majority Voting Scores for different fitness functions

reaches convergence sooner than a 20-agent society does (Figure 5.12). Note the situation at the very end of the society, where there are only two agents remaining. This means that the understanding accuracies that are described in this plot, are accuracies achieved solely on each other's sentences. These accuracies seem to climb linearly as they play language games with one another. This does produce an overfitting effect, as these agents achieve a subpar F-score on the test set (91.0% as opposed to 91.9% for the fittest agent halted at the majority voting halting point).

As we mentioned before, the size, efficiency, understandability and internal consistency fitness function have no positive effect on the evolution of a society from a parse accuracy point-of-view. The other fitness functions however

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
10 Agents	84.8	85.0	92.2	91.8	90.0	90.5	91.8	92.1	91.9	91.5
20 Agents	85.3	85.3	92.0	91.9	91.2	90.9	91.8	92.1	91.6	91.9

Table 5.11: GRAEL-1 - Crossover: Majority Voting Scores for different fitness functions

do optimize the society as a whole in a more guided manner. The F-scores for these fitness functions are less varied than in the previous experiment (circling around 91.9%), but are on the whole also lower. Also, the efficiency and size fitness function seem to hamper the stronger fitness functions in combinations (cf. C1, C3 and C4). The combination of accuracy and understanding accuracy provides a slightly higher F-score, which is however still lower than its 20 agent counterpart.

It is not quite clear which society is to be preferred: the 10-agent society needs fewer language game runs and produces a certain level of stability in the F-scores. It does seem to be outperformed by the 20-agent society, although differences are very small and may be due to random factors.

Sexual Procreation (Crossover)

Neither the results of the 20-agent crossover experiment, nor of the 10-agent splicing experiment differ much from the results reported in Table 5.11. The accuracy fitness function is improved upon, while understanding accuracy loses out. But the combination of the two seems to yield a consistently good result. The combination of the fitness functions have a tendency to neutralize any negative bias (caused by overfitting) either component introduces.

We would like to draw the attention to two peculiar graphs (Figure 5.13). Both plots outline a similar situation: the fittest agent plot is stuck in a local maximum for about 50 language games. Data analysis shows an exceptional situation, apparent in both experiments. The new agents that are created during the creation of the first generation, have a small grammar and have a good internal consistency score (since the probability mass attributed to the grammar rules to parse their own trees is larger). This situation continues in both experiments until the first wave of newborns is finished, after which accuracy scores rise linearly.

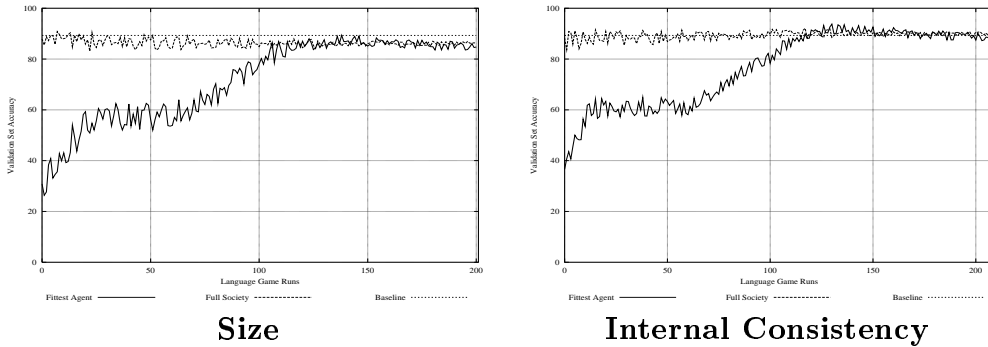


Figure 5.13: GRAEL-1 - 10 agents - Crossover - Size and Internal Consistency Fitness Function

Similar to the 20-agent society, crossover limits the number of language game runs that need to be played, without a loss of accuracy. The C2 fitness function only needs 100 runs in a crossover-based society to achieve an F-score of **92.1%** as opposed to 131 runs in a splicing-based society.

To conclude, there is little to choose between society sizes for crossover experiments. All things being equal, the 10-agent society can be preferred if only for computational reasons. We will argue however that the negative side-effects of convergence in a GRAEL-society makes a larger society size preferable as it provides a larger window of time before convergence, during which the agents can be observed to hold the best grammars (Section 5.2.6).

5.2.3 5 agents

The 5 agent society limits the number of language games that need to be played even further. Since there is little difference between the results of the 20-agent society and the 10-agent society, it might be interesting to see if we can still achieve the same kind of results if we further reduce the number of agents and as a consequence the number of language game runs that need to be played. In a 5-agent society, each agent starts off with a respectable chunk of grammatical information. Fewer agents parse more sentences per language games, which means that convergence happens very fast. This of course influences the performance of the agents in the society. Detailed results for the experiments with 5-agent societies can be found in Appendix

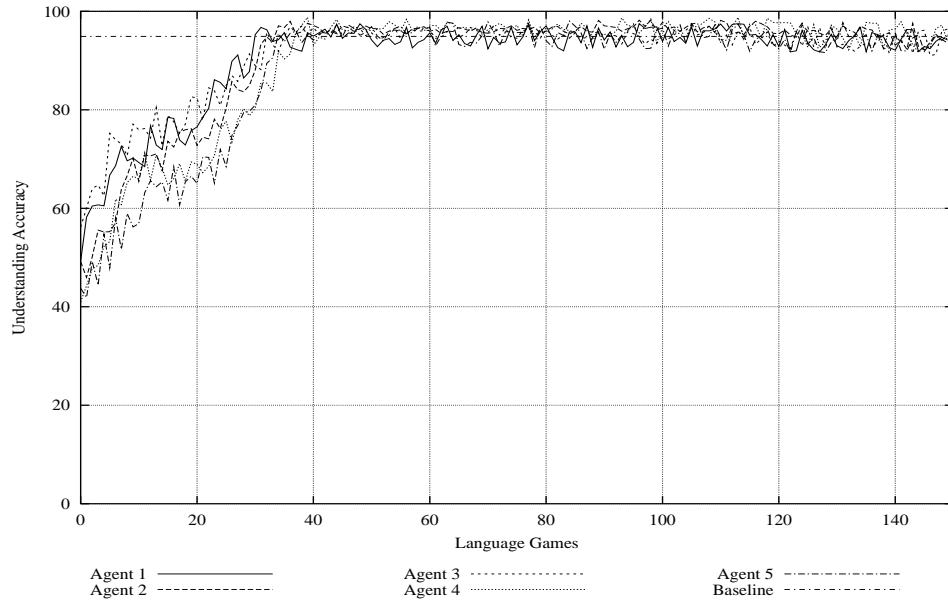


Figure 5.14: GRAEL-1 - 5 agents - Single Epoch - Understanding Accuracy (understanding Fitness Function)

D.

Single Epoch

Figure 5.14 shows that convergence indeed occurs very early: the agents only need about 40 runs to reach the top of the learning curve. After that, understanding accuracy does not increase any more and even seems to decrease after a while.

The F-scores outlined in Table 5.12 suggest that the 5-agent society is marred by serious limitations. It yields underwhelming results compared to the previously described societies. This is caused by the rapid convergence which does not allow the agents to develop fully. Figures 5.3 and 5.11 showed a longer, as well as higher peak moment in the understanding accuracy plots. This peak usually takes place not too long before, or during the moment at which all agents' understanding accuracy start to level out and the society starts to reach a state of convergence. It therefore seems that the more a

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
5 Agents	90.4	90.4	90.5	90.5	90.5	90.7	90.7	90.5	90.7	90.4
10 Agents	90.3	90.8	91.1	91.3	90.3	90.6	91.2	91.3	91.2	91.1
20 Agents	89.9	90.9	91.1	91.0	90.7	90.8	91.1	91.1	91.1	91.1

Table 5.12: GRAEL-1 - Single Epoch: Majority Voting Scores for different fitness functions

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
5 Agents	89.1	91.1	91.2	91.1	90.1	89.3	91.0	91.1	91.0	91.1
10 Agents	82.9	82.9	91.9	91.9	89.6	90.7	91.6	92.0	91.8	91.7
20 Agents	78.5	85.9	92.0	91.7	91.2	90.6	91.6	92.1	91.7	92.0

Table 5.13: GRAEL-1 - Splicing: Majority Voting Scores for different fitness functions

society converges the more the agents’ grammars are reflecting the probabilistic distribution of all the structures in the society, rather than constitute a grammar optimized through “practical” usage, as is the goal of GRAEL-1. The rapid convergence that is apparent in this experiment, however limits the size and duration of this peak, so that it gets harder to find the optimal agent for parsing.

Splicing, Crossover

Figure 5.15 shows the understanding accuracy plots for a GRAEL-society using *understanding* as a fitness function. The introduction of societies delays convergence for a little while, but after about 60 runs, understanding accuracies seem to level. This fast convergence again negatively affects F-scores on the test set, as is evident in Table 5.13. The results of the crossover experiments are scarcely better, as Table 5.14. Even though results are much better than the single epoch experiments, F-scores are well below the 10-agent and 20-agent counterparts, leaving no reason to prefer a 5-agent society, despite its fast development and convergence.

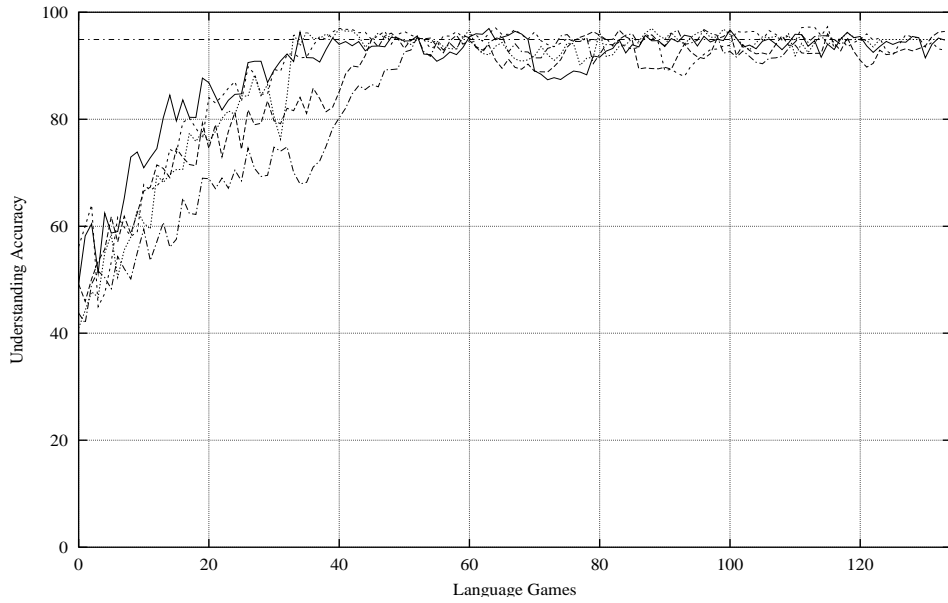


Figure 5.15: GRAEL-1 - 5 agents - Splicing - Understanding Accuracy (understanding Fitness Function)

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
5 Agents	87.3	90.2	91.1	91.1	90.1	88.3	91.0	90.9	91.2	91.2
10 Agents	84.8	85.0	92.2	91.8	90.0	90.5	91.8	92.1	91.9	91.5
20 Agents	85.3	85.3	92.0	91.9	91.2	90.9	91.8	92.1	91.6	91.9

Table 5.14: GRAEL-1 - Crossover: Majority Voting Scores for different fitness functions

5.2.4 50 agents

Let us now look at a larger society of 50 agents, which almost doubles the amount of agents found in the default experiment. This means that 463 sentences are divided over 50 parts, roughly leaving each agent with 10 sentences at the start of the society. Initial F-scores will therefore be low, but especially in generation-based societies, the grammars will be more flexible, as new grammatical information acquired through language games receives a larger portion of the probability mass. Detailed results can again be found in Appendix D.

Single Epoch

The 20 agent single epoch society converged after about 150 language game runs. Using only 10 agents, the number of runs is reduced to 100 runs. Using 5 times as many agents only doubles the amount of runs (200) to reach convergence (see Figure 5.16). Considering the fact that each agent holds fewer sentences and therefore also parses fewer sentences per language game, one might be inclined to think the agent's development would be delayed and that many more language game runs would be required to reach convergence. However, learning is sped up, exactly because each agent holds fewer trees: with a limited initial I-language, agents are not able to parse the other agent's sentences very well, so that the minimal correct substructure (determined by the algorithm described in Chapter 4 (p.104)) that the agents provide to another, are in general more elaborate. The increased amount of knowledge that is being shared helps to counter the delayed development of the society.

The results in Table 5.15 show again that, although the **accuracy** fitness function has proved to be able to produce strong agents, it is also prone to instability. Results show in fact that this fitness function causes the society to halt too early, stranding the society at a point in time at which it has not reached its full potential yet. The **understanding** fitness function, as well as its combination with **accuracy**, however does provide results that compare favorably to the other society sizes. The **efficiency**, **size**, **understandability** and **internal consistency** functions however do not seem to handle this society well at all. They also limit the F-score in combination with other fitness functions.

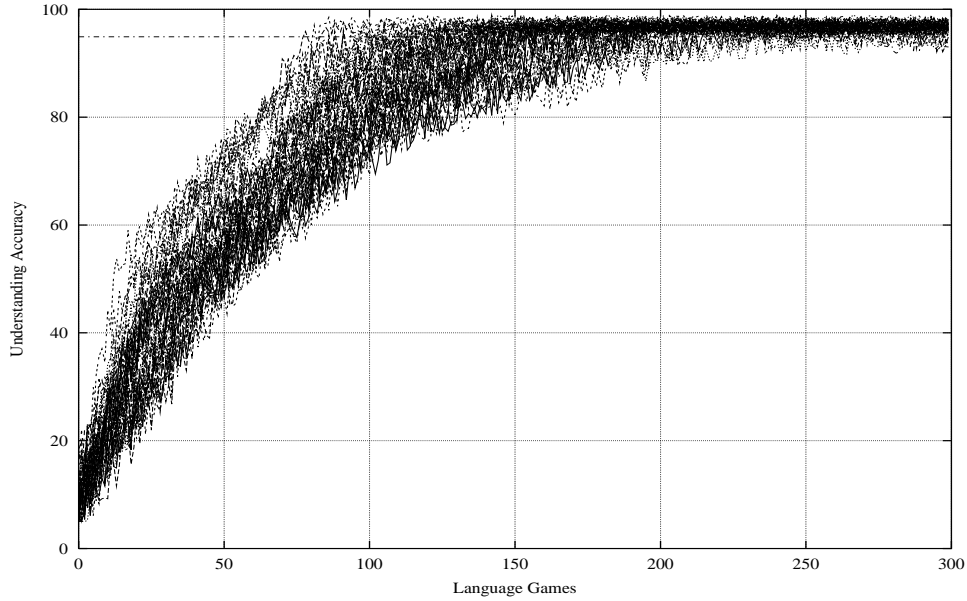


Figure 5.16: GRAEL-1 - 50 agents - Single Epoch - Understanding Accuracy (understanding Fitness Function)

Taking into account the extra CPU-time this larger society entails, it does not seem advisable to use it for single epoch GRAEL-1-societies.

Splicing and Crossover

There are not many surprises to be found in the results of the **splicing** experiments (Table 5.16): the **accuracy** fitness function provides a slight edge when compared to other society sizes, whereas the **understanding** fitness func-

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
50 Agents	90.5	90.0	90.9	91.0	91.0	90.9	90.0	91.1	90.9	90.8
5 Agents	90.4	90.4	90.5	90.5	90.5	90.7	90.7	90.5	90.7	90.4
10 Agents	90.3	90.8	91.1	91.3	90.3	90.6	91.2	91.3	91.2	91.1
20 Agents	89.9	90.9	91.1	91.0	90.7	90.8	91.1	91.1	91.1	91.1

Table 5.15: GRAEL-1 - Single Epoch: Majority Voting Scores for different fitness functions

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
50 Agents	85.8	89.5	92.2	91.3	90.7	88.9	92.0	92.1	92.0	92.0
5 Agents	89.1	91.1	91.2	91.1	90.1	89.3	91.0	91.1	91.0	91.1
10 Agents	82.9	82.9	91.9	91.9	89.6	90.7	91.6	92.0	91.8	91.7
20 Agents	78.5	85.9	92.0	91.7	91.2	90.6	91.6	92.1	91.7	92.0

Table 5.16: GRAEL-1 - Splicing: Majority Voting Scores for different fitness functions

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
50 Agents	82.1	81.6	92.1	91.9	89.7	90.2	91.8	92.2	91.9	92.0
5 Agents	87.3	90.2	91.1	91.1	90.1	88.3	91.0	90.9	91.2	91.2
10 Agents	84.8	85.0	92.2	91.8	90.0	90.5	91.8	92.1	91.9	91.5
20 Agents	85.3	85.3	92.0	91.9	91.2	90.9	91.8	92.1	91.6	91.9

Table 5.17: GRAEL-1 - Crossover: Majority Voting Scores for different fitness functions

tion loses out severely. This is due to some unfortunate halting points that eschew the decision of the majority voting procedure. Because not more than 15 runs later, the F-score is significantly higher at **92.1%**.

As a consequence, the combination of accuracy and understanding is not affected with a 92.1% F-score. And the combination of understanding with understandability, size and efficiency, which indeed seem to slow down the development of the society, produces better results than the isolated understanding fitness function. In fact, all combinations of fitness functions seem to work well in this society size.

The **crossover** experiment results are similar, apart from the fact that this method again reduces the number of language game runs significantly. In this experiment the combination of understanding and accuracy again proves to be the best fitness function to use, with a **92.2%** F-score. All combinations of fitness function produce respectable results, as do the single accuracy and understanding fitness functions. Apart from the idiosyncratic behavior of the understanding fitness function in the splicing experiment, the 50-agent society seems to yield F-scores that are on the whole more stable and robust than in the other society sizes.

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
100 Agents	90.5	90.9	91.1	90.9	89.8	87.6	90.7	91.0	90.7	91.0
5 Agents	90.4	90.4	90.5	90.5	90.5	90.7	90.7	90.5	90.7	90.4
10 Agents	90.3	90.8	91.1	91.3	90.3	90.6	91.2	91.3	91.2	91.1
20 Agents	89.9	90.9	91.1	91.0	90.7	90.8	91.1	91.1	91.1	91.1
50 Agents	90.5	90.0	90.9	91.0	91.0	90.9	90.0	91.1	90.9	90.8

Table 5.18: GRAEL-1 - Single Epoch: Majority Voting Scores for different fitness functions

5.2.5 100 agents

Finally, we turn to society size with 100 agents. This is a very large society, especially given the rather limited size of the corpus: each agent will hold a minimal number of four to five sentences. This means that the number of language game runs will again need to be larger to reach convergence, but also that the structures that are shared during the initial language games are larger as well. With such a high number of possible candidates for language games, and such little information to guide the agents at the onset, the initial stages of these societies will largely be guided by chance. Detailed results of experiments with the 100-agent societies can be found in Appendix D.

Table 5.18 shows that the 100-agent single-epoch society compares well to the other society sizes. Again, the strongest contenders are the accuracy and understanding fitness function and the combination of the two. Notice the subpar scores for the understandability and internal consistency fitness functions. With each agent’s E-language minimal in size, these fitness functions are very vulnerable, as the grammatical information of the E-language has little or no impact on the I-language.

The results of the experiments with generations (Tables 5.19 and 5.20) corroborate previous claims. The accuracy fitness function performs well in larger societies, while the understanding fitness function trails a little (which is especially true for the crossover experiment). But the combination of the two does not fail to score the best among all fitness functions.

The 100-agent societies produce results that are not subpar, but at no point rival the other society sizes in performance. Combined fitness functions seem to suffer from the effects of the society size, which makes the results

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
100 Agents	90.1	90.3	92.0	91.8	90.0	90.3	90.5	91.8	92.0	91.9
5 Agents	89.1	91.1	91.2	91.1	90.1	89.3	91.0	91.1	91.0	91.1
10 Agents	82.9	82.9	91.9	91.9	89.6	90.7	91.6	92.0	91.8	91.7
20 Agents	78.5	85.9	92.0	91.7	91.2	90.6	91.6	92.1	91.7	92.0
50 Agents	85.8	89.5	92.2	91.3	90.7	88.9	92.0	92.1	92.0	92.0

Table 5.19: GRAEL-1 - Splicing: Majority Voting Scores for different fitness functions

	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
100 Agents	87.2	86.9	92.0	90.6	88.4	88.0	92.0	92.0	91.9	91.2
5 Agents	87.3	90.2	91.1	91.1	90.1	88.3	91.0	90.9	91.2	91.2
10 Agents	84.8	85.0	92.2	91.8	90.0	90.5	91.8	92.1	91.9	91.5
20 Agents	85.3	85.3	92.0	91.9	91.2	90.9	91.8	92.1	91.6	91.9
50 Agents	82.1	81.6	92.1	91.9	89.7	90.2	91.8	92.2	91.9	92.0

Table 5.20: GRAEL-1 - Crossover: Majority Voting Scores for different fitness functions

less stable. There is therefore no added advantage to using this society size, especially considering the extra computational costs it entails.

5.2.6 General Comments

Explaining Shapes

Understanding Accuracy plots such as Figures 5.3, 5.11 and 5.14 allow us to look at the course of the experiments, purely in terms of the degree to which agents are able to parse each other’s parse forests. For different society sizes, as well as different generation methods, there are some general tendencies noticeable in the way a society develops over time.

Figure 5.17 shows understanding accuracy plots for the single epoch experiment. The plots were bezier smoothed to more clearly indicate general tendencies in the data. Looking at the overall plot, we can distinguish three stages: (1) an acquisition phase during which the agents are building their grammars, while the plots linearly increase (runs 0 - 100 in Figure 5.17). This

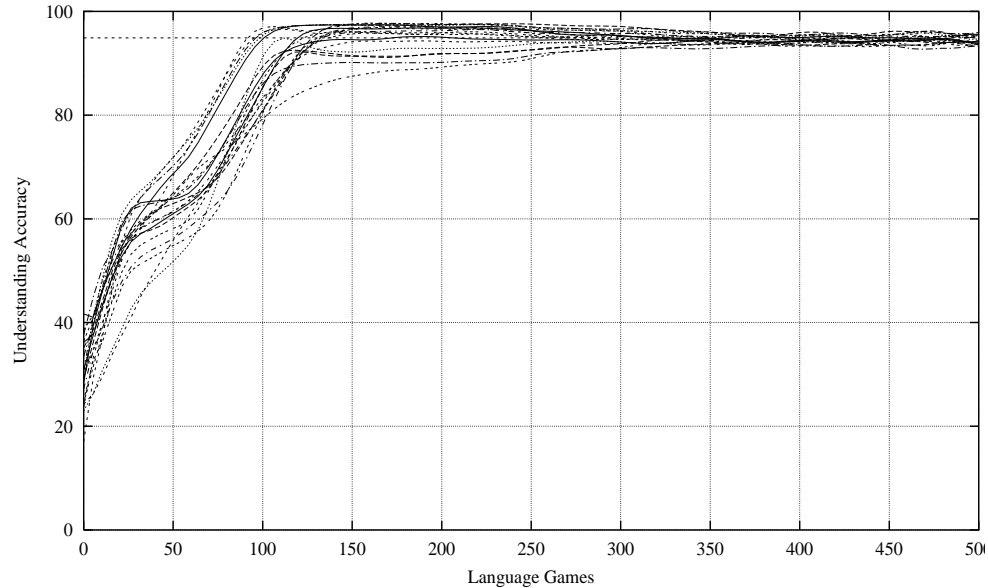


Figure 5.17: GRAEL-1 - 20 Agents - Understanding Accuracy (understanding Fitness Function) - Smooth Bezier Curves

is followed by (2) a peak phase during which the accuracies of the most understanding agents are at their highest (runs 100 - 150) and a (3) convergence phase, during which the agents' adapt to one another's grammar (runs 150 - 500), arriving at a situation where most agents' grammars are very similar to one another.

The acquisition phase (1) can be further subdivided into roughly two or three more parts: (i) a step phase during which the agents learn nothing but new structures (1-25), an adjustment phase (ii) during which the agents' accuracies appear to level (25-50) for a short period of time and an optimization phase (iii) during which at the same time, more new grammatical structures are acquired and grammatical structures acquired in (i), (ii) and (iii) are statistically optimized (50-100). Phase (ii) is exceptionally pronounced in Figure 5.17, but seems to be mostly absent in other experiments. The subdivision of the acquisition phase seems only relevant for single epoch experiments, because the agents' plots in generation-based experiments are marked by periodical decreased accuracy in newborn agents. It is phase (ii) that often causes halting procedures such as plateau detection to prematurely

halt at a local maximum.

An important observation is that the last phase (3), during which the agents' grammars converges is seemingly characterized by a steady decline of understanding accuracy. Figure 5.17 illustrates this effect: as the convergence phase progresses, the better agents start to score slightly lower understanding accuracies, while the lesser agents' accuracies climb. It is during the relatively short phase preceding the convergence phase (2) that we want the society to halt, because the experiments have shown that halting points located in this phase produce stronger agents than halting points in the convergence phase. We would like to dub this the stage of *beneficial confusion*: the agents do not fully agree on a grammatical system yet and the distribution of the probability mass reflects actual usage in a practical context, rather than constituting an even allocation over a common set of grammar rules, mirroring the original data set.

It is important to point out that the society as a whole does not deteriorate during phase (3): the agents become very much similar to one another, which weakens the better agents and improves the worse agents. As opposed to many other agent-based implementations, convergence in the GRAEL-1 society is not a desirable situation, since the pre-convergence situation produces the better parsers. A GRAEL-1 society should therefore make sure that the stage of beneficial confusion last long enough to allow for a halting procedure to trigger a proper halting point. We have found that larger population sizes in general do indeed provide a longer beneficial confusion stage, which makes them preferable for GRAEL-1 processing, unless we are sure that our halting procedure is trustworthy.

Halting Procedures

The discussion of the difference phases again brought up the importance of the halting points. So far, we have mainly discussed the majority voting method. The majority voting halting procedure was defined as follows: if at least 4 out of 7 halting procedures have suggested a halting point prior to or at the present point in time, the majority voting procedure halts the society. This means that the majority voting halting point is identical to at least one other halting point.

	FS vs TS	FA vs TS	FA vs FS	Plateau Detect.	Under- Stand.	Limit LG	Limit Gen.	Major. Voting
Total	22	47	51	37	53	45	6	69
Unique	10	5	10	9	27	26	4	0

Table 5.21: Halting Points - Best Result

But is majority voting really the best halting procedure? Table 5.21 provides an overview of the number of times each halting procedure produced the best result. The majority voting clearly has the upper hand, even though it is only the best solution in less than half the experiments. Luckily, in the other cases, the F-score the majority voting method yields is however rarely significantly lower than the other best halting procedure: in 102 experiments the difference was less than 0.2%.

Data analysis showed that very rarely does the majority voting halting procedure halt the society at the same time as the understanding accuracy halting procedure does. In 125 of the 150 experiments though, the understanding accuracy halting point occurs **before** the majority voting halting point, indicating that it more often than not plays a role in triggering the majority voting procedure. Table 5.21 now also suggests that this halting procedure is indeed not much worse at determining the halting point than the majority voting procedure is. Furthermore, it does not need extra processing in the way the other halting procedures do. It would be interesting to see if we can squeeze more performance out of this particular halting procedure. More generally, it would also be interesting to implement weighted voting, so that some halting procedures have a bigger weight towards the final decision of the majority voting than others.

5.2.7 Summary of Results and discussion

Let us now summarize the results. Since the combination of the understanding and accuracy fitness functions consistently provided top-of-the-line results, we will base our comparison on these. Table 5.22 provides a general overview of the results: if for some reason, we do not want to use a generation-based GRAEL-society the 10 agent society is the obvious choice. In any case, even without introducing new generations, GRAEL-1 outperforms the base-

Method	Size	Halting Point	$F_{\beta=1}$ -score (%)
Single Epoch	5	47	90.5
	10	107	91.3
	20	107	91.1
	50	233	91.1
	100	240	91.0
Splicing	5	63	91.1
	10	131	92.0
	20	237	92.1
	50	266	91.9
	100	342	91.8
Crossover	5	94	90.9
	10	100	92.1
	20	195	92.1
	50	234	92.2
	100	333	92.0
Baseline	1	0	89.3

Table 5.22: GRAEL-1 ATIS - Majority Voting - understanding+accuracy Fitness Function - Comparison

line method by a significant margin.

Agents in a generation-based society will disappear from the society, while, depending on their fitness, part of their grammatical information lives on in their offspring. Since only a certain percentage of the grammar is transferred to the newborn agents, the use of generations is in effect a way to purge the grammar of lower frequency rules, which means that they constitute rules, rarely used during language games. This provides an important performance increase over the single epoch experiments. There is little to choose between the two methods for creating new generations on the basis of their accuracy on the test set. The crossover method however significantly reduces the number of language game runs, while providing a (albeit non-significant) performance increase for some experiments over splicing.

The 10-agent society seems to be preferable as it only needs 100 runs to

Single Epoch	Baseline	Crossover	Baseline	Crossover	Single Epoch	Crossover	Splicing
40	19	36	23	27	15	9	33
4	433	6	431	31	423	35	419
$P(8.5) = 0.003$		$P(8.8) = 0.003$		$P(4.9) = 0.03$		$P(0.001) = 0.9$	

Table 5.23: GRAEL-1 ATIS - McNemar Tests for 10-agent Societies

achieve a **92.1%** F-score. This excellent result can be attributed to a well-balanced set of halting points, which triggers the majority voting halting point at an ideal point in time. If we are insecure about the efficiency of the halting point and we would like to extend the stage of beneficial confusion, the 20-agent is a good fail-safe choice, as it provides a much larger window in time, at which the F-scores are peaking, so that we need less “luck” to halt the society at a good point.

The 50-agent crossover society provides an insignificant .1% advantage over the other society sizes. Combined fitness functions are more stable in this society size. This is especially important if we wish to limit the amount of parsing by eliminating the validation set. Using only understanding, understandability, efficiency and size, the society is able to produce an agent scoring **92.0%** on the test set (using splicing).

Considering the 10-agent society as the preferred society, we can calculate significance scores using McNemar tests. Table 5.23 shows that the difference in recall between the single epoch society and the baseline is significant. Naturally, the result of the crossover experiment is significantly different from the baseline accuracy, but also from the single epoch society. The difference in recall between the two generation methods is not significant.

5.2.8 Extra experiments

Before we turn to the WSJ-experiments, we want to discuss some minor extra experiments that provide some extra insights and/or serve as a sanity check to the previously described experiments.

Random Crossover

One experimental parameter we discussed was the **random crossover** operation. Random crossover allows two agents to exchange structures outside of the context of a language game. This operation can speed up convergence, since more grammatical information is doing the rounds, but the question remained whether it would negatively affect performance.

The thought exercise in Appendix C helped us to limit the number of experiments that needed to be done, by only testing those combinations of experimental parameters that were deemed relevant. We proposed that the random crossover parameter would only be affected by the method of creating new generations and the population size. We can therefore limit the experiments to just one fitness function (**understanding+accuracy** is the logical choice), one halting point (majority voting) and one data set (ATIS).

Random crossover was implemented as follows: after each language game run, each agent was linked to another agent, after which they start exchanging grammar rules at random: one structure for each sentence in the E-language. There is only one condition on the exchange: the nodes need to carry the same top-level node label¹².

Table 5.24 shows for the three generation methods the difference in results between the default setting (**no RC**) and the setting that enables random crossover. We notice that the number of language game runs is indeed reduced significantly: since grammatical information is distributed more quickly throughout the society, convergence occurs sooner, as the agents will sooner have all possible grammatical constructs at their disposal. The effect is more noticeable in the generation-based experiments when agents reach end-of-life sooner.

Not surprisingly perhaps, random crossover introduces a significant performance decrease for almost all societies. With the exception of the oddball 5-agent society, all GRAEL-1 perform significantly worse. The effect is not as pronounced in the splicing experiment: the use of generations is intended to prune useless grammatical information, so that the detrimental influence of the random crossover is somewhat diminished. But the generation method based on crossover, in which newborn agents inherit on average more gram-

¹²Regardless of the index provided by PMPG.

	Single Epoch				Splicing				Crossover			
	No RC		RC		No RC		RC		No RC		RC	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
5 Agents	47	90.5	41	90.4	63	91.1	51	91.0	94	90.9	71	91.1
10 Agents	107	91.3	89	90.6	131	92.0	97	91.7	100	92.1	82	91.4
20 Agents	107	91.1	101	90.6	237	92.1	174	91.8	195	92.1	187	91.5
50 Agents	233	91.1	178	90.5	266	91.9	201	91.7	234	92.2	188	91.6
100 Agents	240	91.0	203	90.5	342	91.8	267	91.5	333	92.0	248	91.5

Table 5.24: GRAEL-1 ATIS - The Effect of Random Crossover on Test Set F-score

grammatical information than newborns out of splicing, is harmed more severely.

In conclusion: even though random crossover does speed up convergence, it significantly affects accuracy scores for almost all types of GRAEL-1-societies. It therefore seems pointless to implement this extra step in inter-agent communication, especially given the insight that the agents do not benefit from entering a state of convergence.

Varying Inheritance Ratio

The experiment with random crossover raises a question on the actual difference between the two generation methods. Even though McNemar tests find no significant difference in the results between these two methods, there are enough indications that they do have a different effect on the performance of a GRAEL-1 society. We already noted that newborn agents from a crossover operation have on average a larger initial I-language than newborn agents that are the product of splicing. This can explain why crossover reduces the number of language game runs: newborn agents have a larger I-language and therefore need less grammatical information to reach the end-of-life state, hence new generations are created in shorter intervals.

But the question remains whether it is just the size of the initial I-language that causes the agents to require fewer language games to replenish their grammars, or if it is also related to the actual way in which it is compiled, i.e. from two agents as opposed to just one agent. We therefore conducted two experiments: one experiment in which the splicing method provided the newborn agents with a larger I-language, and one experiment in which the crossover method provided a smaller I-language to newborns (Figure 5.18).

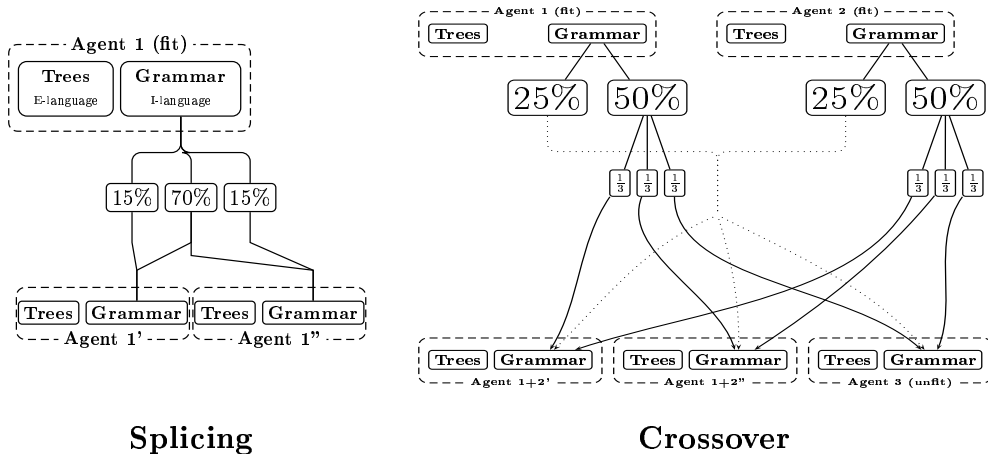


Figure 5.18: Updated thresholds for Generation methods

Both experiments were conducted on a 20-agent society, using the majority voting halting procedure and the understanding+accuracy combination as a fitness function.

Table 5.25 shows the results of the experiments with the new thresholds for determining the newborn agent's I-language size. In the new splicing experiment, a newborn agent holds 85% of its ancestor's grammar, as opposed to 75%. This is roughly similar to a newborn agent's size in a 20-agent crossover society. The number of language game runs decreases, but not by much, while the F-score on the test set also seems to suffer. This may have something to do with the splicing method: as opposed to the crossover method, the splicing method does not let any of the ancestor's I-language go to waste. The common part between the two newborn agents is larger, making their I-languages more similar than in the default experiment. This undoubtedly speeds up convergence, which, as we have proposed before, is not necessarily a good thing.

The new settings for the crossover experiment causes newborn agents to have a smaller grammar than before: about 25% of the ancestor's I-language is just thrown away. This does not seem to hurt the society as a whole, though, since there is hardly any difference in the amount of language game runs it takes the society to reach the majority voting halting point. There is a slight decrease in F-score, but the difference is probably not significant. It does however indicate that the reduction of language game runs in a crossover

	Default		Updated	
	LGR	F	LGR	F
Splicing	237	92.1	225	91.9
Crossover	195	92.1	200	92.0

Table 5.25: Updated Threshold for Generation Methods - Experimental Results

experiment is not solely due to the size of the newborn agents' I-language. We believe the difference to be mainly due to a more advantageous method of creating the initial I-language, i.e. by culling structures from two agents, rather than just one.

This experiment is also illuminating in that the new settings for the crossover experiment do not decrease performance, nor decrease or increase the number of language game runs. It therefore seems to be a more stable method for creating new generations, i.e. less vulnerable to the influence of different threshold setting. And its most significant change, i.e. the creation of smaller newborn agents, may in fact have a beneficial effect on parsing times as well. Future research should look into ways to determine the optimal settings for the generation methods.

Fitness Functions for Procreation vs Fitness Functions for Selection

The main difference between the single epoch experiments and the generation-based experiments is the fact that in the former, fitness functions are only used to select the fittest agent in the society, and therefore do not have an effect on the course of the experiments as such. In generation-based experiments, fitness functions are used to select agents for procreation (for convenience sake we will dub this kind of selection SELPRO), as well as select the fittest agent that needs to parse the test set (henceforth SELFIT). These are however different types of selection and it may be interesting to see whether there are any significant differences to be observed when we alternate SELPRO and SELFIT. In other words: we define two different types of fitness: fitness for procreation and fitness for parsing.

The 20 agent crossover experiment is the basis for our comparison, with

SELFIT→ SELPRO↓	SI	EF	AC	US	UB	IC	C1	C2	C3	C4
SI	85.3	86.3	88.6	87.9	85.9	84.2	86.2	88.2	87.0	86.9
EF	84.2	85.3	89.6	88.3	85.9	84.9	85.9	90.0	88.9	88.2
AC	90.2	91.5	92.0	92.0	91.1	91.9	91.5	92.2	92.0	91.7
US	90.3	90.2	92.0	91.9	91.5	90.6	91.8	92.0	92.0	92.0
UB	90.0	91.2	91.3	91.3	91.2	91.2	91.3	91.4	91.1	91.1
IC	89.3	90.0	91.2	91.1	90.5	90.9	91.0	91.5	91.3	91.2
C1	90.4	91.0	92.0	91.9	91.2	91.7	91.8	91.8	91.5	91.8
C2	90.1	91.3	92.2	92.1	90.9	91.2	91.8	92.1	92.0	91.9
C3	89.8	91.5	91.9	91.8	91.6	91.2	91.2	91.8	91.6	91.8
C4	90.0	90.2	91.9	91.8	91.2	91.0	91.6	91.9	91.9	91.9

Table 5.26: Fitness Functions for Procreation vs Fitness Functions for selection

F-scores based on the majority voting halting points. Note that this does not require extra experimental runs: SELPRO has already determined the course of the experiments. All we need do is apply difference SELFITS on the same experimental run.

Table 5.26 describes the results of this experiment: the figures in bold (diagonal) are the F-scores from the default experiments (with SELPRO = SELFIT). Size is not very well suited as a SELPRO: it produces F-scores that are well below the baseline, even when coupled with more powerful SELFITS such as accuracy. As a SELFIT itself, it seems to have a tendency to select the worst agent of the society. More or less the same conclusion can be drawn with regards to efficiency, even though the effect is not as pronounced. Sometimes efficiency achieves respectables F-scores as a SELFIT, but as a SELPRO it is subpar.

Accuracy, understanding and the combination of the two (C2) as a SELFIT seems to consistently select the fittest agent in a society, no matter how bad results in general are. Note that understanding as a SELFIT rarely performs worse than accuracy. So even though accuracy has the edge over understanding as a SELPRO, they produce similar results as a SELFIT. This experiment provides more evidence for the previously made claim that the selection of agents made by the understandability and internal consistency fitness functions, is unrelated to the agents' quality as a parser for unseen data. As a SELPRO they achieve mediocre results, that only scarcely outperform single epoch experiments, and as a SELFIT their performance is erratic, ranging

from bottom- to top-of-the line scores.

All in all, there is little advantage to be gained by applying a different SELPRO and SELFIT to a society. There are some instances where performance is definitely improved, but applying a different SELFIT to a society guided by a strong SELPRO does not yield an important performance increase.

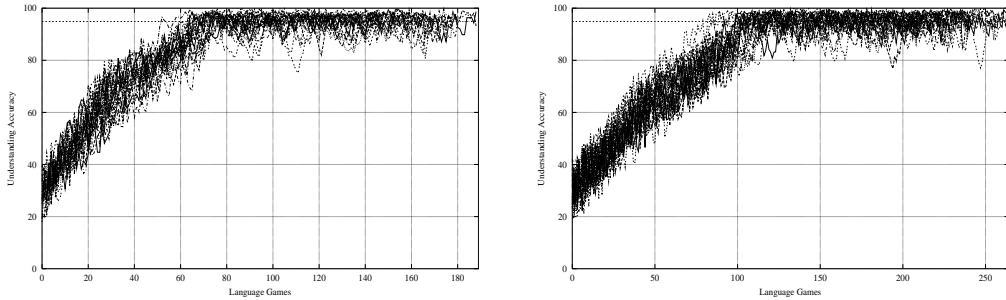
I-language vs E-language

In Chomskian linguistics, E-language is defined as the set of externalized utterances a language user is able to produce. I-language on the other hand is language knowledge internalized by language users. To produce and understand utterances (E-language) we apply the knowledge present in the I-language. The concepts of I-language and E-language in the GRAEL-system are used in a different sense: E-language is a fixed set of utterances an agent is able to produce, and the underlying syntactic structures. The I-language, initially induced from the structures in the E-language, is used to interpret, i.e. parse, another agent's E-language.

We have already discussed the lack of interaction between the I-language and E-language on page 132. In language games, agents acquire new grammatical information, enlarging the I-language, while the E-language remains constant. At no point in the GRAEL-1 experiments do agents use the newly acquired grammatical information to update their E-language, nor do they produce new, previously unseen utterances, on the basis of their I-language.

Since it is not the goal of the GRAEL-1 experiments to provide a psycholinguistically realistic approximation of a human language user's brain, but rather to provide the best possible agent-based method for grammar optimization, this rigid boundary between I-language and E-language is justifiable.

It could however be interesting to see what happens if we allow some interaction between these two components. We tried two approaches: the first experiment allowed the agents to parse their own sentences after each language game run using their newly updated I-language and replace the structures in their E-language with those parses. The 2nd experiment allowed the agents to randomly generate syntactic structures from their I-language



I-language parses E-language **I-language generates E-language**

Figure 5.19: Interaction between I-language and E-language - Graphs

in a top-down fashion.

The two experiments were conducted on a 20-agent society with sexual procreation and using `understanding+accuracy` as a fitness function and majority voting as a halting procedure. Figure 5.19 displays the course of both experiments. The first experiment shows a standard plot, except that F-scores are reaching 100% many times. This occurs in the convergence phase, when the agents' grammars are becoming more and more similar to one another. It is therefore possible that the structures in one agent's E-language are identical to the structures that another agent produces for these sentences. The other experiment looks very similar, even though there is a totally different society underneath the hood. At some points, understanding accuracies do reach 100%, but since the structures in the E-language are generated by the I-language, the results are more dispersed.

Table 5.27 displays the results of these experiments. At the majority voting halting point, the fittest agent in the society was selected and its grammar was used to parse the test set. The F-score for the first experiment is still quite respectable, but there is a significant decrease compared to previous societies. The F-score for the 2nd experiment is lower still, even though it is still higher than baseline accuracy.

It is interesting to note that removing the barrier between I-language and E-language does not send the GRAEL-1 society into disarray. But if we want to create an agent to parse unseen data, the GRAEL-1 society benefits from maintaining the original tree-structures from the original corpus. Experiments with GRAEL-2 (Chapter 7) however will show that the strict

	F-score
Baseline	89.3
No Interaction	92.1
I-language parses E-language	91.8
I-language generates E-language	90.9

Table 5.27: Interaction between I-language and E-language - Experimental Results

Baseline PMPG	89.3
GRAEL-1 PMPG	92.1
Baseline PCFG	83.6
GRAEL-1 PCFG	86.0

Table 5.28: GRAEL-1 scores with a PCFG

distinction is not useful in a grammar induction task.

Aspects of Pattern-Matching

The parser that agents used in the GRAEL-1 experiment is the PMPG-method described in Chapter 3. It tries to mimic a human language user's predilection for larger substructures in parsing. Also, knowledge-sharing between agents in the GRAEL-system is based on this idea: the algorithm for finding the minimal correct substructure (Chapter 4, p. 104) focuses on substructures of trees, rather than single rewrite rules.

To check if the performance increase GRAEL-1 produces is not some side-effect of the PMPG-method, we also ran a GRAEL-1 experiment using a simple PCFG as a parser. Again we used a 20-agent crossover society, using `understanding+accuracy` as a fitness function and majority voting to determine the halting point. Table 5.28 shows the result of this experiment. A GRAEL-1 system powered by a PCFG does increase significantly over the baseline model, although it is lower than the PMPG baseline accuracy. Furthermore, the PMPG-based GRAEL-society yields an error reduction rate of 26.2%, while the PCFG GRAEL-society only achieves a 14.6% error reduction rate.

	Correct	LP		LR		$F_{\beta=1}$ %
		/	%	/	%	
Total	4412	4765	92.6 (± 0.9)	4819	91.6 (± 4.3)	92.1 (± 2.5)

Table 5.29: 10-fold Cross Validation Experiment for GRAEL-1

These results indicate that GRAEL-1 does optimize grammars, even when using a different parser, but PMPG intrinsic preference to larger substructure seems more suited to the dynamics of the GRAEL-system and the algorithm that selects the minimally correct **substructure** in particular.

Sanity Check: 10-fold cross-validation

One final experiment on the ATIS-corpora checked whether the improvement GRAEL-1 achieves can be attributed to a favorable partitioning of the original corpus. Using a different seed, we randomly divided the ATIS-corpora again into 10% partitions of equal size. Each partition was used as a test set and validation set once, while the other eight partitions served as the training corpus for the GRAEL-society.

For reasons of time, we used a 10-agent crossover-based society. **understanding+accuracy** was the fitness function of choice, while majority voting decided on the halting point. Table 5.29 displays the results for the 10-fold cross validation experiment. The F-scores range from 89.8% to 94.7%. The overall F-score is **92.1%** which is not significantly different from the GRAEL-1 results previously described.

5.2.9 Some Details about the Data

It is clear that GRAEL-1 indeed optimizes grammars for parsing. This is done by providing a society of agents with a “deficient” grammar and allowing them to improve on it by “practicing” their own grammars on other agents. By helping each other out, the grammars become optimized in a setting which resembles the actual task at hand: parsing sentences. In the end, the agents in the society have acquired a grammar, which is superior to the grammar

that was used to create the society. Whereas in standard parsers, like the ones described in Chapter 3, the distribution of the probability mass is based on each constituent’s frequency in the training set, GRAEL-1 redistributes this mass to reflect optimal use for parsing.

So far, we have provided a lot of quantitative data about the GRAEL-1 experiments, but have not really looked underneath the hood of the system to see what exactly is going on. In this section, we look at the data generated by agents in a GRAEL-society.

Grammar Optimization: an example

First we look at a typical example of how GRAEL-1 is able to overcome difficulties of the baseline model. Let us consider the parse for the sentence: “*What flights go from Tampa to Charlotte on Sunday*” in Figure 5.20¹³. This sentence, which is troublesome for a simple PCFG due to its embedded VP, is also not handled very well by a PMPG. Figure 5.21 shows the PMPG-parse for this sentence. The highly idiomatic WHNP-construct with an empty PP is found, as well as the entire VP. The parse is however far from correct, mainly because of the erroneous attachment of the empty particle.

Using a parser powered by a grammar induced from the fittest agent from any of the better performing GRAEL-1 generation-based societies however produces the correct analysis in Figure 5.20. When we look at the language games in the initial stages of the society, we notice that the erroneous attachment of the empty element, made by PMPG in Figure 5.21 is a very frequent mistake, because of the relatively high frequency of WHNP-constructs with an empty PP. It is therefore normal, at least initially, for an agent to propose analyses with such WHNP-structures. As the society progresses, agents that suggest tree-structures of the type featured in Figure 5.21, will be provided with a minimal correct substructure like the one in Figure 5.22.

The parsing agent will include this substructure in his I-language. This step may need to be repeated several times in subsequent language games with other agents, but finally, this structure will obtain a probability in the grammar such that it overcomes the over-eager pattern-matching that is

¹³We have added words to this structure for clarity’s sake, but actual parsing was done on part-of-speech tags.

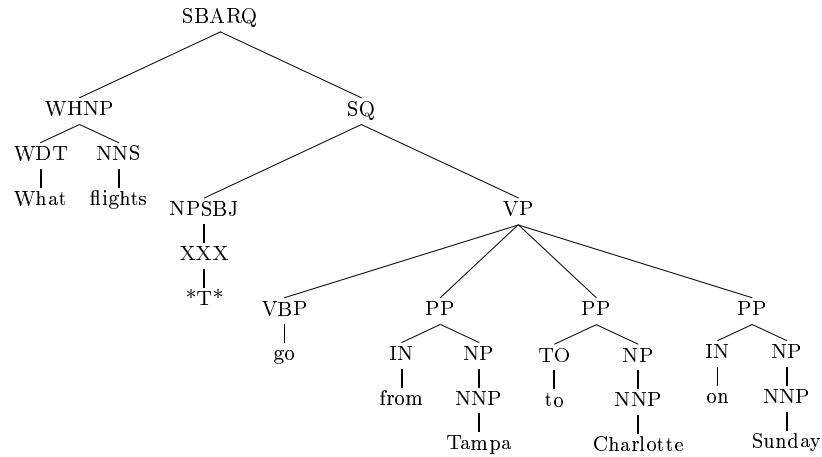


Figure 5.20: Correct Parse for “What flights go from Tampa to Charlotte on Sunday”

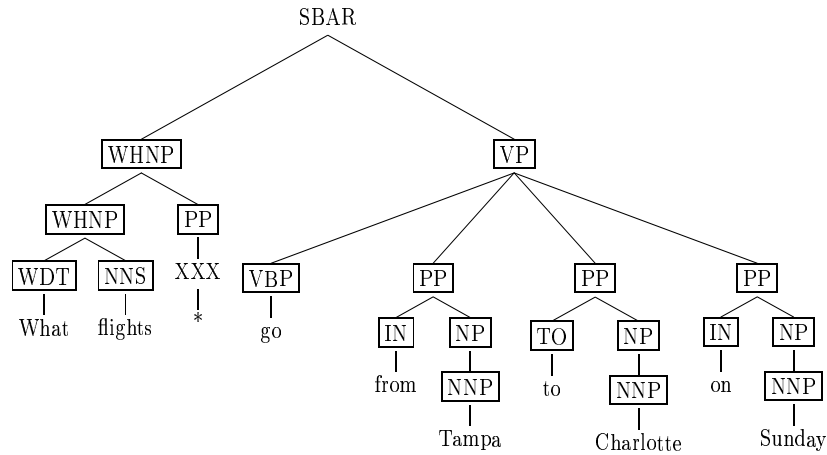


Figure 5.21: PMPG parse for “What flights go from Tampa to Charlotte on Sunday”

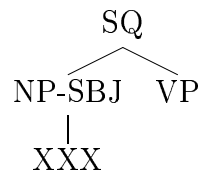


Figure 5.22: Minimal Correct Substructure

evident in Figure 5.21. In other words: the increased probability it gains, nudges the probabilistic component of the PMPG to consider the SBARQ-analysis over the highly matchable SBAR-analysis.

This example provides a good illustration of how GRAEL-1 optimizes grammars. It fine-tunes probabilistic grammars such that the probabilities are suited to actual parsing. In this view, annotated corpora are viewed as unoptimized raw data that constitute building blocks for a grammar, which are still lacking fine-tuned probabilities. GRAEL-1 provides a way to approach the ideal distribution of probability mass for the task of parsing unseen data.

Size and Efficiency Fitness Functions

We introduced fitness functions in Chapter 4 as a possible method to obtain grammars that are enhanced for a specific kind of purpose. Size and efficiency fitness functions, for instance, can be used to try and build a small and fast grammar for parsers. So far we have mainly evaluated GRAEL-societies by looking at their performance on the test set. Let us now take a look at the fitness functions of size and efficiency to see what kind of grammar these fitness functions produce in a GRAEL-society. The data discussed in this section stems from the 20-agent crossover society, except for the last two lines in Table 5.30.

Turning back to the results in Table 5.26 allows us to inspect the size and efficiency functions more closely. This table illustrated that they constitute bad SELPROS and seemed to have a tendency to select some bottom-of-the-line agent as a SELFIT.

Table 5.30 describes for a number of SELPRO-SELFIT combinations, the number of (unique non-indexed) rules in the grammar¹⁴, the F-score and the CPU-time¹⁵ used to parse the test set. We notice that the efficiency function is able to reduce CPU-time significantly. As a SELFIT for a society based on the understanding+accuracy fitness function, it provides an agent that reduces the CPU-time by almost 30% compared to the default SELFIT, albeit with a significant accuracy drop. It is also interesting to note that efficiency in

¹⁴The actual indexed grammar (on which size-fitness is based) can be huge. We do not provide these figures as they are to a large extent based on chance.

¹⁵Figures express an average of 5 runs on a Dual AMD Athlon1.2Ghz CPU.

SELPRO	SELFIT	rules	$F_{\beta=1}$	CPU-time
Size	Size	251	85.3	49 sec
Size	US+AC	255	88.2	51 sec
Efficiency	Efficiency	263	85.3	43 sec
Efficiency	US+AC	261	90.0	45 sec
US+AC	US+AC	285	92.1	70 sec
US+AC	Size	273	90.1	63 sec
US+AC	Efficiency	283	91.3	51 sec
C2(50)	C2(50)	299	92.2	71 sec
C4(50)	C4(50)	279	92.0	60 sec

Table 5.30: size and efficiency Fitness Functions

general produces larger grammars than size, but not at the cost of CPU-time, illustrating the previously made claim that there is no one-to-one relationship between a grammar's size and its efficiency.

The last two lines of Table 5.30 describe data from the 50-agent society. The understanding+accuracy(C2) society (both SELFIT and SELPRO) yields an accuracy score of 71 seconds. The combination of all fitness functions (C4), which includes both size and efficiency still obtains a top-of-the-line F-score, coupled with a 15% decrease in processing time.

So even though the size and efficiency fitness functions have a negative influence on the performance of a society as a SELPRO as well as on its fittest agents as a SELFIT, it is interesting to note that if practical reasons require the grammar to be efficient, the GRAEL-society is able to deliver.

This concludes the description of the GRAEL-1 experiments on the ATIS dataset. So far the results indicate that GRAEL-1 provides a workable grammar optimization method. Given the limited size of ATIS, however, there is a danger that differences in results that appear significant, are actually the result of (un)favorable conditions. We will now turn to experiments on the WSJ-corpus to see if we can corroborate claims made with regards to the ATIS-experiments on a large-scale corpus.

5.3 Experimental Results: the WALL-STREET-JOURNAL Corpus

The WSJ-corpus holds 40.000 sentences, which makes it impossible for present-day technology to apply the GRAEL-system on the entire corpus. A 20-agent society would need to generate 114.000 parses¹⁶ using the entire WSJ-corpus. With a conservative estimate of 1 minute parsing time per sentence for the WSJ-corpus, a 200 language game run experiment would last 16 years on a single computer.

We therefore conducted the main WSJ-experiments on a 1% subsample of the WSJ-corpus consisting of 1.000 sentences. Parsing times are much lower and fewer structures need to be parsed per run, so that a WSJ-experiment can be over in a couple of weeks, rather than years. Once the experiments on the subset are over, we decide on the best suited GRAEL-society to process the full corpus, an experiment described in Section 5.3.4.

As in the ATIS-experiments, we conduct experiments on five different population sizes. But this time, we only experimented on two fitness functions: the *understanding* fitness function and the *understanding+accuracy* fitness function. The former experiment is used as a test to see how well GRAEL performs without the use of a validation set. To further reduce processing times, halting procedures that require the use of validation set were not used for this experiment. All halting procedures were used for the experiment with the *understanding+accuracy* fitness function, which has proved to be the most stable fitness function in the ATIS-experiments.

We use sexual procreation to create new generations, as this has proved to reduce the number of language game runs without loss of accuracy. We also showed that its performance seems less subject to threshold settings that determine the size of newborn agents' grammars.

One final note on the parser used: whereas we used the integrated model for the ATIS-experiments (see Chapter 3, p. 76), we use the optimized method, proposed in Chapter 3 (p. 62), of using a PCFG to generate the

¹⁶32.000 during language games, 20x4.000 validation set sentences by agents, 1x4.000 validation set by full society.

n most likely parses for each sentence¹⁷, after which PMPG applies its re-ranking scheme on the parse forest. This has the possible disadvantage that a correct parse generated by the parser may not be featured in the miniature 20 sentence parse forest the PCFG outputs, so that not even the PMPG re-ranking scheme can salvage the parse. Previous experiments on the WSJ-corpus showed however that **97%** of the time, the correct parse can be found among the 10 most probable parses in the ordered parse forest generated by a PCFG. The test set however was parsed by the fittest agent in the society using the integrated method generating a full parse forest.

5.3.1 20 agents

We start off again with the default society size of 20 agents. 800 sentences of the training set are distributed over 20 agents, while 100 sentences serve as a validation set and 100 more as a test set. Detailed results of the experiments can be found in Appendix E.

Understanding

The course of the experiment, plotted in Figure 5.23 runs very similar to the ATIS-experiments, except that initial understanding accuracy in language games is slightly higher but the peak is slightly lower. The first remarkable thing is that the WSJ-experiment does not need many more language game runs than the ATIS-experiment (342vs332). This is not as strange as it seems, since in this experiment, each agent holds many more sentences in its E-language, so that more information is passed per language game run. This also causes rapid convergence after 150 runs. The crossover operation apparently makes sure that agents are born with decent I-languages, so that agents reach end-of-life at regular intervals.

The fittest agent plot in Figure 5.24 is also similar to the ones found in the ATIS-experiments. The main difference is the F-score, which is much lower. Baseline accuracy (full training set used to parse the validation set) is at 79.9%. The fittest agent overtakes the baseline model around the 150th run. Table 5.31 shows the results for the halting procedure that halts the

¹⁷ n was set to the arbitrarily chosen value of 20 for the GRAEL-1 experiments.

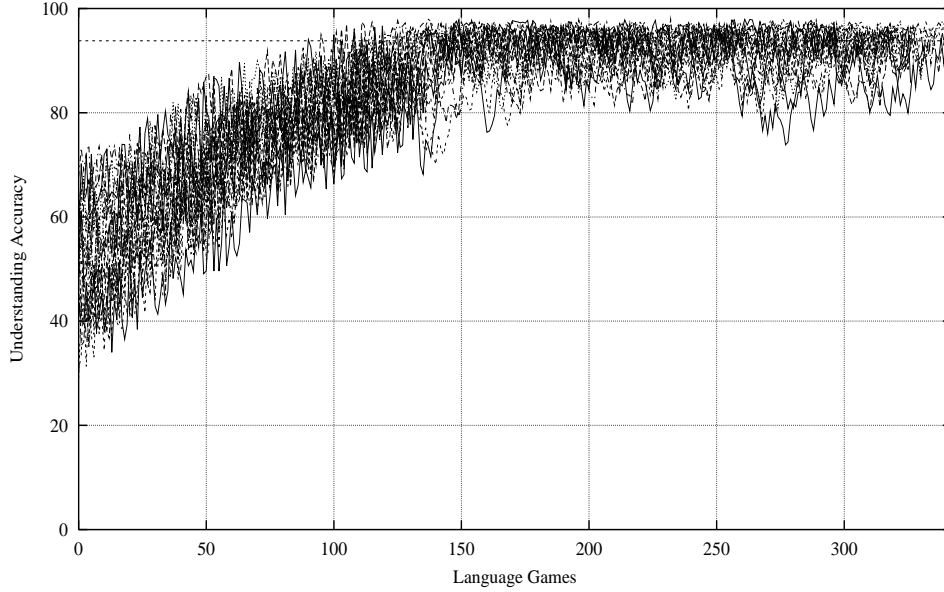


Figure 5.23: GRAEL-1 WSJ - 20 Agents - Crossover - Understanding Accuracy (understanding Fitness Function)

	Halting Point	$F_{\beta=1}$
Baseline	0	79.9
20 Agents (US)	152	81.2

Table 5.31: GRAEL-1 WSJ - Results

society at the point where understanding accuracies are levelling out. We do not consider majority voting for the experiment with the understanding fitness function, since it does not use most of the halting procedures. The fittest agent in the society at the understanding halting point achieves **81.2%** F-score on the test set. Exact Match accuracy for GRAEL-1 is 26%, while the baseline only scores 20.0%. Both the difference on Exact Match accuracy and the F-score are significant according to the McNemar test (Table 5.32).

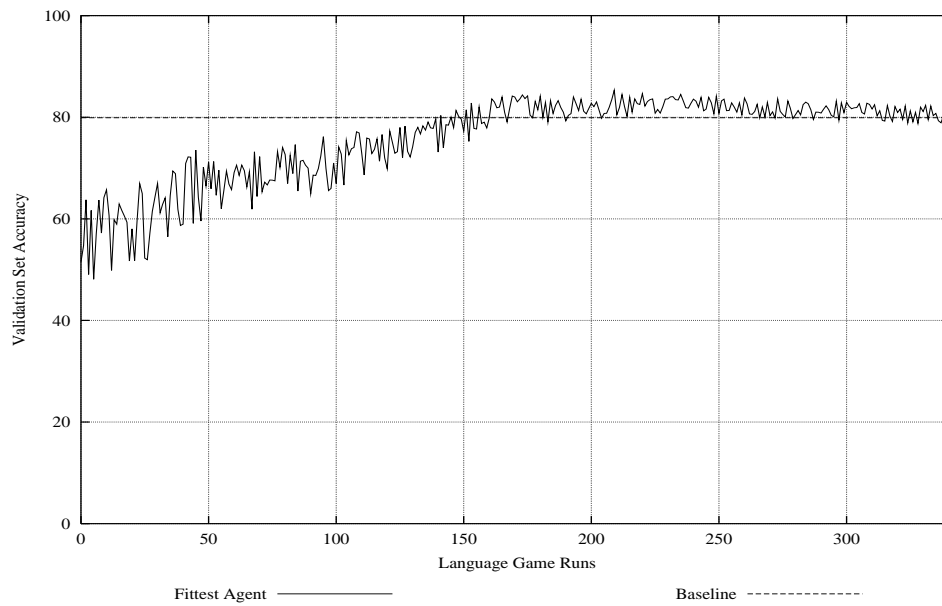


Figure 5.24: GRAEL-1 WSJ - 20 Agents - Crossover - Fittest Agent Accuracy vs Baseline Accuracy on Validation Set(US Fitness Function)

	Exact Match		Constituents	
baseline accuracy vs	74	6	270	172
GRAEL-1 accuracy	0	20	134	1623
	$P(4.2) = 0.041 < 0.05$		$P(4.5) = 0.034 < 0.05$	

Table 5.32: McNemar Tests for GRAEL-1 WSJ experiment

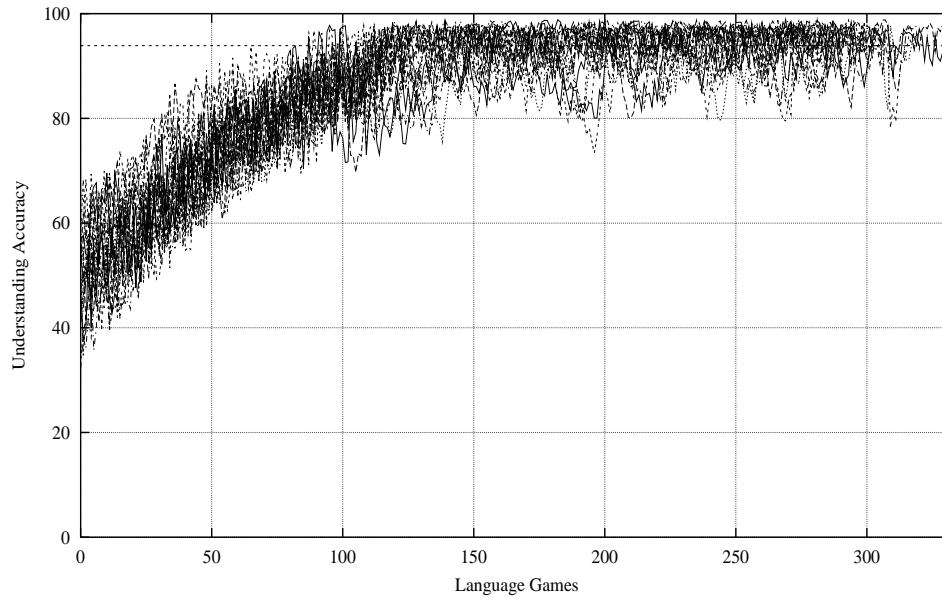


Figure 5.25: GRAEL-1 WSJ - 20 Agents - Crossover - Understanding Accuracy (C2 Fitness Function)

Understanding+Accuracy

The next experiment uses the combination of understanding and accuracy as the operative fitness function in the GRAEL-1 society. This combination has proved to consistently provide the best results during the ATIS-experiments. Since parsing the validation set is part of the fitness function, we can consider all halting procedures for this experiment.

Figure 5.25 displays the course of the experiment. It is very similar to that of Figure 5.23 except that overall understanding accuracy seems to be a bit higher, which can be attributed to the stronger fitness function. This is also noticeable in the fittest agent plot in Figure 5.26. The fittest agent halted at the majority voting halting point also produces a slightly better result than in the previous experiment (Table 5.33).

The 20-agent experiment described so far support the grammar optimization skills of GRAEL-1, even though observed differences seem smaller. The main problem seems to be that each agent holds many more sentences

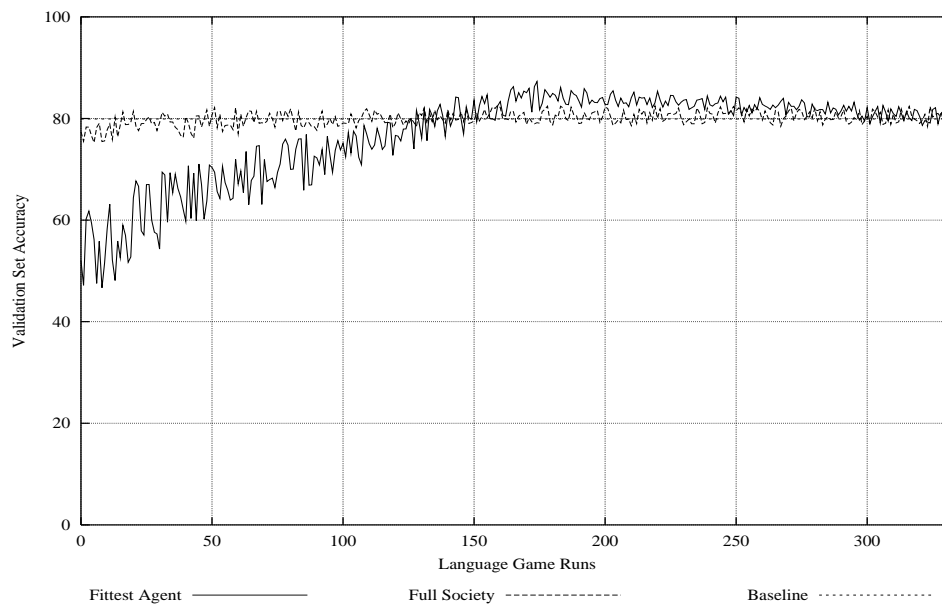


Figure 5.26: GRAEL-1 WSJ - 20 Agents - Crossover - Fittest Agent Accuracy vs Full Society Accuracy vs Baseline Accuracy on Validation Set (C2 Fitness Function)

	Halting Point	$F_{\beta=1}$
Baseline	0	79.9
20 Agents (US)	152	81.2
20 Agents (C2)	184	81.4

Table 5.33: GRAEL-1 WSJ - Results

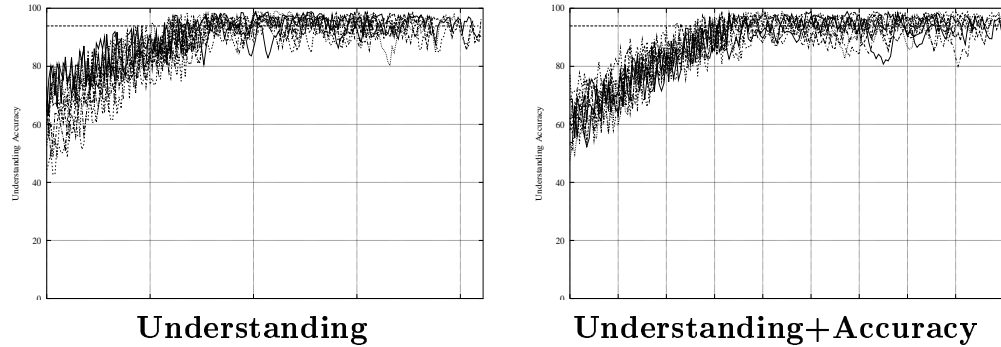


Figure 5.27: GRAEL-1 WSJ - 10 Agents - Crossover

compared to agents in an ATIS GRAEL1-society, creating a situation that is similar to the 5-agent GRAEL-1 ATIS-experiment, during which convergence happened too fast and individual agents held too much grammatical information, so that the information acquired during language games had a harder time acquiring a substantial enough portion of the probability mass. When the state of beneficial confusion (during which we wish to halt the society) is short, like in the WSJ-experiments described so far, agents have very little time to achieve such a redistribution before getting dragged along into the slipstream of the convergence phase. On a more positive note, we would like to remark that even though the difference over the baseline seems much smaller, the McNemar tests show that the difference observed so far is easily just as significant as in the ATIS-experiments.

5.3.2 10 agents, 5 agents

Given the fact that the 20-agent society seems to be marred by the limited number of agents present, there is little hope for the smaller society sizes. We therefore shortly deal with them in one combined section. The details of these experiments can be found in Appendix E.

Figure 5.27 shows the course of the two 10-agent experiments. Whereas the understanding+accuracy experiment yielded slightly higher understanding accuracies, there is hardly any distinguishable difference between either method in the 10-agent society. The results show that the 10-agent society is indeed able to reduce the number of language games, but each fitness

	Halting Point	$F_{\beta=1}$
Baseline	0	79.9
20 Agents (US)	152	81.2
20 Agents (C2)	184	81.4
10 Agents (US)	122	81.1
10 Agents (C2)	118	81.2
5 Agents (US)	72	80.0
5 Agents (C2)	100	80.4

Table 5.34: GRAEL-1 WSJ - Results

function now scores lower than in the 20-agent society. The aforementioned problem of agents containing too much grammatical information is even more apparent in this experiment.

The fittest agent plots for the 5-agent society (Figure 5.28) show that the agents start off with a high accuracy on the validation set and that there is little left to gain by playing language games with other agents. Even though GRAEL-1 surpasses baseline accuracy, the result is still significantly lower than that of the larger society sizes. This experiment again corroborates the claim that a larger corpus needs a larger number of agents, whose initial grammars are smaller and therefore more easily tweakable in the GRAEL-society.

5.3.3 50 agents, 100 agents

By increasing the number of agents in the society and thereby providing each agent with an initial I-language that is much smaller in size compared to the I-languages at the onset of the 20-agent society, we hope to alleviate some of the aforementioned problems. Details of the 50-agent and 100-agents experiments can be found in Appendix E.

It is clear from Figure 5.29 that these experiments run a similar course to one another, regardless of the fitness function used. Even though the accuracies of the 100-agent society are more scattered, they do converge along the same lines, with only the number of language game runs as a

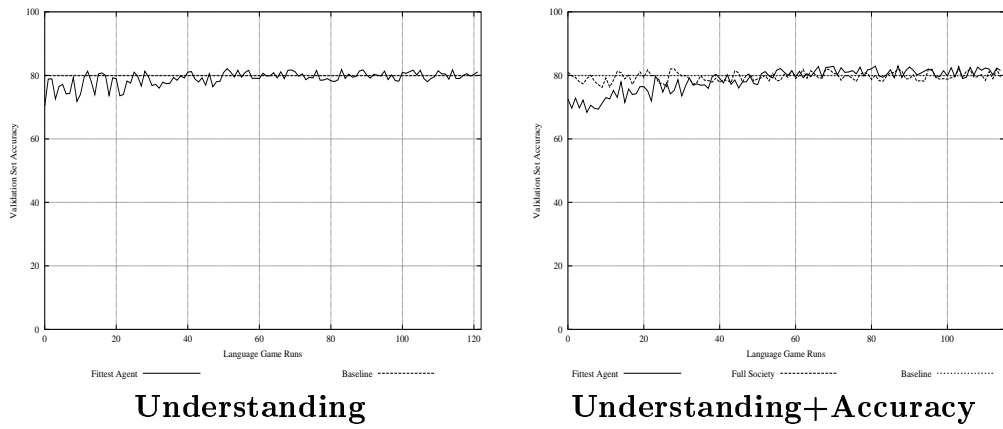


Figure 5.28: GRAEL-1 WSJ - 10 Agents - Crossover - Fittest Agent Accuracy (vs Full Society Accuracy) vs Baseline Accuracy on Validation Set

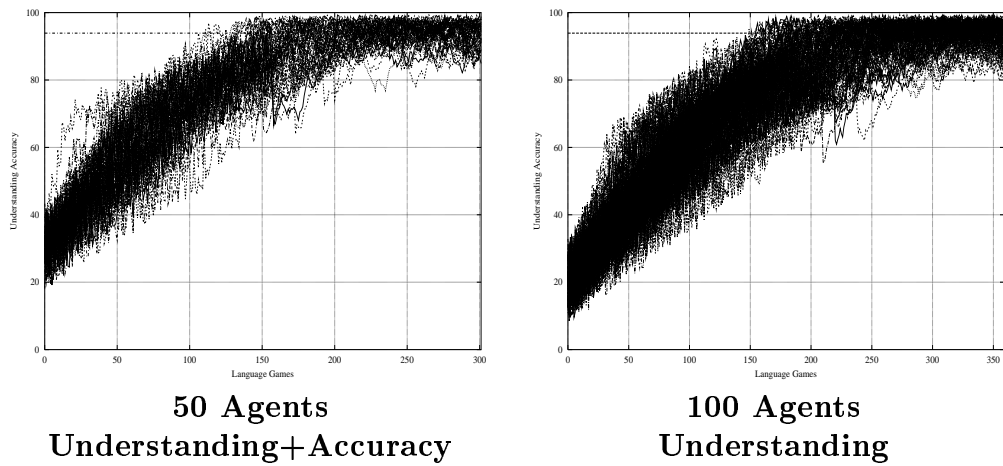


Figure 5.29: GRAEL-1 WSJ - 10,50 Agents - Crossover

	Halting Point	$F_{\beta=1}$
Baseline	0	79.9
100 Agents (US)	255	81.4
100 Agents (C2)	260	81.4
50 Agents (US)	175	81.3
50 Agents (C2)	223	81.4
20 Agents (US)	152	81.2
20 Agents (C2)	184	81.4
10 Agents (US)	122	81.1
10 Agents (C2)	118	81.2
5 Agents (US)	72	80.0
5 Agents (C2)	100	80.4

Table 5.35: GRAEL-1 WSJ - Results

distinguishable difference. The 50 agents society improves its performance on the respective fitness functions over the 20 agent GRAEL-1-society. The observed differences are significant over the baseline model, as well as over the societies with fewer agents.

The 100 agents society does not improve its figures over the 50 agents society, but helps to illustrate the point that a larger corpus requires a larger population size. The fact that a 100 agent society does not yield a large improvement over 50 agents, however seems to indicate that no more improvement can be expected by further increasing population size. Doing so would only increase the number of language game runs.

These experiments complete our results Table 5.35. The society size of choice appears to be the 50-agent society, as it provides a favorable balance between the number of language game runs and the F-score achieved on the test set.

5.3.4 The Main Experiment

We now need to decide on a GRAEL-1 instantiation to apply on the full WSJ-corpus. It is computationally unfeasible to use the standard GRAEL approach

of using a validation set to determine halting points and/or determine the fitness of agents, as it more than doubles the amount of parsing in each language game run. Table 5.35 however suggests that, if our population size is large enough, the difference between the two fitness functions largely disappears.

This means we can reduce processing times by abandoning the validation set and by focusing on settings that only operate on knowledge obtained during language games. But even without processing on a validation set, one single GRAEL-1 experiment using the full WSJ-corpus would be computationally intractable, as there are 40k sentences in the standard WSJ training set (partition 2 to 21)¹⁸. It is also not clear what the population size should be for a 40k training set and we currently lack the resources to dynamically find out.

We therefore propose the following adjustment: the 40.000 sentences of the training set (partitions 2 to 21) are divided into 40 parts of 1.000 sentences each. At the onset of the GRAEL-society, 1 of the 40 partitions is distributed over the agents. Every 10 language game runs, all agents' E-languages are purged and replenished with the next partition of 1.000 sentences. The agents' I-language is also enriched with the structures from the newly acquired E-language. This method does not need to disturb the development of an agent to any great extent: every 10 language game runs, it will need to adapt to a new set of sentences, but it will also receive new grammatical information to do so.

This method makes sure that each sentence in the corpus is featured in the society at one point or another. Since new grammatical information is added to the society all the time, we need a new method to determine when an agent has reached end-of-life and is ready to procreate. Based on observations made with respect to generation-intervals on the standard GRAEL-1 experiments, we implemented the following approximation for determining the lifespan of an agent:

First Generation	Consecutive Generations
$n * 2 + \text{rand}(n)$	$n + \text{rand}(n)$
$n = \text{the number of agents in the society}$	

¹⁸In our experiments, partition 22 is used as an optional validation set and partition 23 is used as the test set.

The first generation of agents is allowed to build up its grammar for a longer period of time to establish a firm grammatical basis for future generations. Consecutive generations occur randomly with a minimal interval of n runs, with n being defined as the number of agents in a society.

We decided to conduct two experiments: one with a 50 agent society and one with a 100 agent society. Some minor adjustment were made to speed up processing:

- Understanding is the SELPRO in both experiments
- A specialized version of `understanding+accuracy` is used as a SELFIT: if the halting point has reached the top 5 most `understanding` agents in the society are asked to parse the validation set. The agent that obtains the highest F-score is then selected to parse the test set.
- Only two halting points are considered: the plateau in understanding accuracy and the very end of the society, i.e. when all 40 partitions have been used

The graphs in Figures 5.31 and 5.30 show a course of events that is similar to the experiments with the ATIS-corpus and the restricted WSJ-corpus, which is encouraging. The corpus seems to be diverse enough not to hurt agents in their development, if new information is introduced and the approximation of GRAEL-1 seems to run a similar course to the standard instantiation.

Figure 5.31 shows some downward, as well as upward peaks every 10 runs at the start of the society. This means that some agents' understanding accuracy is stunted by the new wave of sentences, while others clearly benefit from the added advantage of having their I-language enriched with new grammatical structures.

The 50-agent society is halted after the 312th language game run. At this point the specialized SELFIT procedure selects the fittest agent in the society to parse the test set. It achieves a **80.7%** F-score, while the 100-agent society improves on that even further with **81.1%**. Exact Match accuracies are also greatly improved over the baseline model. We will show in Chapter 6 that this is a consequence of GRAEL's preference for larger substructures.

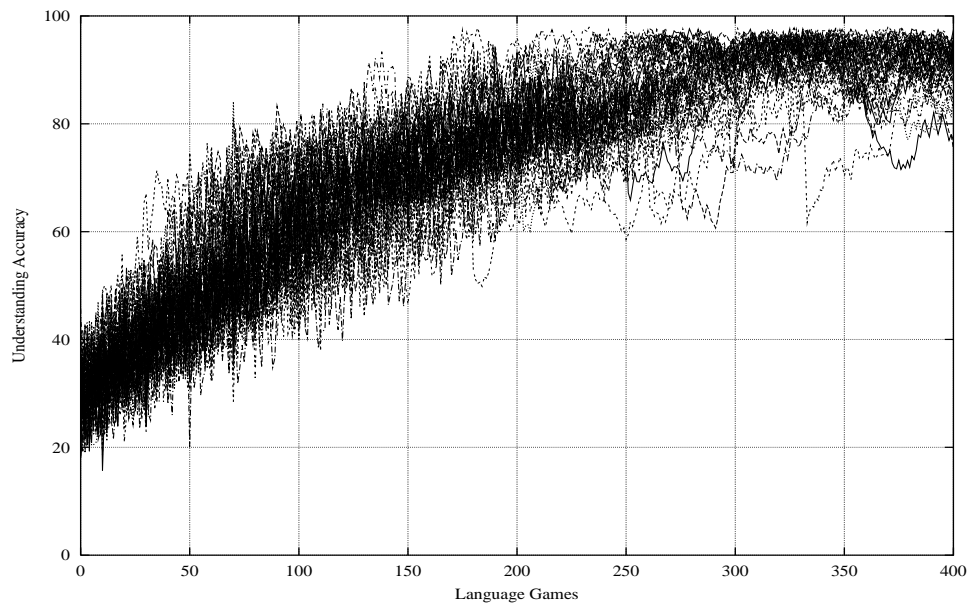


Figure 5.30: GRAEL-1 WSJ - 50 Agents - Final Experiment - Understanding Accuracy

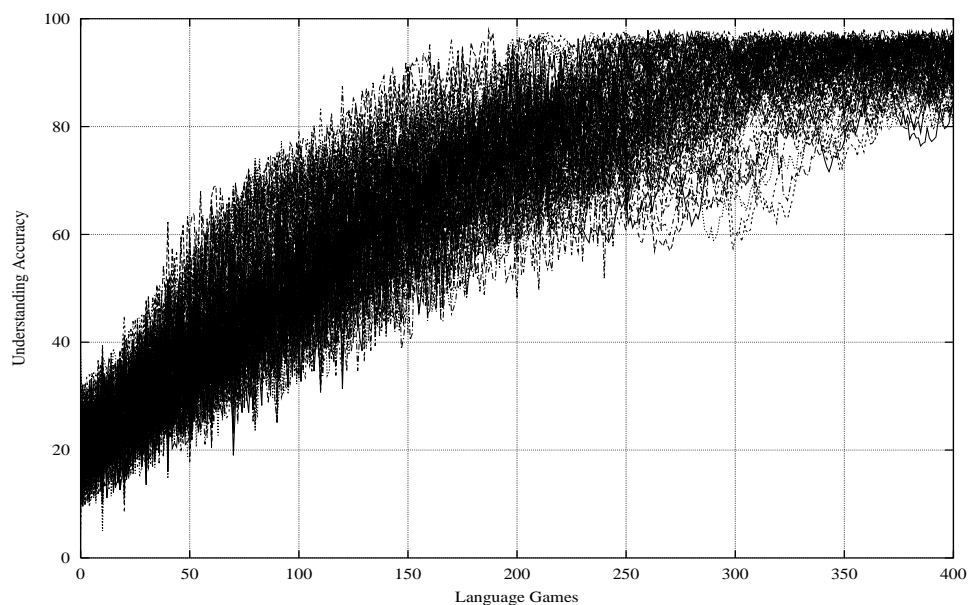


Figure 5.31: GRAEL-1 WSJ - 100 Agents - Final Experiment - Understanding Accuracy

	Exact Match		Correct	LP		LR		$F_{\beta=1}$ %
	/2416	%		/	%	/	%	
PMPG	386	16.0	37535	45908	47333	81.8	79.3	80.5
50 Agents	537	22.2	38195	46904	47333	81.4	80.7	81.1
100 Agents	551	22.8	38401	46985	47333	81.7	81.1	81.4

Table 5.36: wsj - Final Experiment - Results

	Exact Match		Constituents	
baseline accuracy vs GRAEL-1 50 Agents	1792	238	1411	8387
	87	299	7727	29808
	$P(69.2) = 8.7E - 17 < 0.05$		$P(26.9) = 2.1E - 07 < 0.05$	
baseline accuracy vs GRAEL-1 100 Agents	1784	246	1598	8200
	81	305	7334	30201
	$P(82.3) = 1.2E - 19 < 0.05$		$P(48.2) = 3.9E - 12 < 0.05$	
GRAEL-1 50 Agents GRAEL-1 100 Agents	1593	286	7541	1597
	272	265	1391	36804
	$P(0.3) = 0.58 > 0.05$		$P(14.1) = 0.0002 < 0.05$	

Table 5.37: McNemar Tests for Final GRAEL-1 wsj experiments

Thanks to the large test set, we are finally able to calculate significant tests on a substantial data sets for experiments with GRAEL-1 (Table 5.37). The results of the 50-agent, as well as the 100-agent GRAEL-1 society are significantly different from those of the baseline model. Even though in absolute terms the difference in for instance F-score may not be as big as in the ATIS-experiments, the improvement on the wsj-corpus is much more significant. The difference between the 50-agent and the 100-agent society is only significant on the constituent level.

The experiments with the wsj-corpus have given more evidence for the ability of GRAEL-1 as a grammar optimization method. The final wsj-experiment allowed us to back up this claim with some robust statistical evidence. It should moreover be noted that this experiment was only an approximation of the actual GRAEL-1-method. It is not clear whether the absence of consistent processing on a validation set and the rotation of agents' E-language hurts performance to any great extent. It will be interesting to see how close our approximation has come to actual GRAEL-1 processing on

large datasets, once technology has advanced enough to make the experiment computationally tractable.

5.4 Advances and Future Work

GRAEL-1 provides an interesting method for grammar optimization. It supports the view that an annotated corpus by itself can be considered as raw material in need of optimization. The grammar induced from the annotated sentences does not readily hold the best distribution of probability mass to parse the sentences themselves. GRAEL-1 optimizes a grammar by breaking it down and distributing it over a group of agents. While these agents practice their grammars on one another, they learn to distribute the probability mass over the grammar in a setting that is an explicit reflection of the task they were set out to accomplish. We define GRAEL as an agent-based/distributed evolutionary computing approach for grammar optimization. The agent-based aspect of GRAEL allows for several alternative grammars to be developed simultaneously, while the evolutionary aspects of the environment make sure only those alternatives survive over time that comply with the fitness functions we apply to the society.

There is however a fine balance between the number of agents and the size of the initial corpus. Agents should not hold too many sentences (e.g. the 5-agent ATIS experiment), as this provides them with an initial grammar that is too *set in its ways* to be enhanced by inter-agent knowledge sharing. But a society of agents that start off with a very small I-language (cf. the 100-agent ATIS experiment) seems to produce grammars that are mainly driven by grammatical structures acquired in language games, which also seems to reduce the top parsing accuracy such a society can achieve.

The experiments with GRAEL-1 not only researched grammar optimization in an agent-based environment, but also served as a trial run for the next set of experiments. One of the more desirable traits of GRAEL is its ability to turn faulty grammars into good grammars, simply by having them interact with each other. This would seem very practical as an unsupervised method for grammar induction, which is typically marred by limited performance (cf. Chapter 8). And if we tone down some of the aspects of GRAEL-1 that are explicitly geared towards grammar optimization, such as

the explicit knowledge sharing and the strict distinction between I-language and E-language, the basic sensibilities paramount to GRAEL-1 may provide a setting to attempt a computational simulation of the origins of compositional language (Chapter 10).

Before we turn to these experiments however, we will first compare GRAEL-1 to similar methods for grammar optimization in Chapter 6. In Chapter 7, we will then extend GRAEL-1's data-driven grammar optimization skills, by introducing a mutation operator that is able to create new grammatical information, while the generation-based approach makes sure only useful structures are maintained, projecting GRAEL as grammar rule discovery method.

Gentlemen, when two separate events occur simultaneously pertaining to the same object of inquiry we must always pay strict attention.

Agent Dale Cooper - Twin Peaks, Season I (ep.4) - ©1990

6

A comparison between GRAEL-1 and Ensemble Learning Techniques

An important trend in the field of Machine Learning sees researchers employing combinatory methods to improve the classification accuracy of their algorithms. Natural language problems also seem to benefit from the combination of classifiers to deal with the large datasets and expansive array of features that are paramount in describing this difficult and disparate domain which typically features a considerable amount of sub-regularities and exceptions.

Not only system combination and cascaded classifiers [van Halteren et al. 1998; De Pauw and Daelemans 2000; Tjong Kim Sang et al. 2000] are well-established methods in the discipline of Machine Learning for natural language, also ensemble learning techniques such as **bagging** and **boosting** have been applied successfully on a number of natural language classification tasks [Abney et al. 1999; Hoste and Daelemans 2000; Henderson and Brill 2000]. These techniques hold in common that in no way do they alter the actual content of the information source of the predictor. Simply by re-

distributing the data, different resamplings of the same dataset are generated to create a combination of complementary classifiers.

In Chapter 5, we described experiments with GRAEL-1 in which we took a corpus of tree-structures and distributed them evenly over a society of agents. Through an extended series of language games, the agents were able to restock their initial faulty grammars with grammatical information in a way that optimizes the distribution of the probability mass for parsing. Simply by re-distributing the data over a group of agents (cf. bagging) and adjusting the weights of particular subsets of the information involved (cf. boosting) GRAEL was able to produce a classifier which outperformed the initial dataset, without changing the actual content of the information source. The strong similarities between GRAEL and the ensemble learning methods of bagging and boosting, warrants a direct comparison.

To our knowledge, only [Henderson and Brill 2000] has so far attempted to apply the methodology of bagging and boosting on treebank parsers, even though [Collins 2000a] describes a method for reranking parse forest using similar techniques. The concepts and experimental setup outlined in [Henderson and Brill 2000] will be the basis for the experiments in this chapter. We will introduce the methods of bagging (Section 6.1) and boosting (Section 6.2) and outline the respective similarities with the GRAEL-system. The experimental comparison and some concluding remarks are presented in Sections 6.3 and 6.4.

6.1 Bagging

The basis idea behind ensemble learning techniques is to collect a number of different classifiers for a particular task, each with their own systematic bias towards classification and combine the results into one classifying combination that ideally incorporates all of the strengths of its parts and none of the weaknesses. Trained on different data sets or powered by different machine learning algorithms, each classifier may be better at some subset of the classification problem than others. If the individual classifiers complement each other sufficiently, an ensemble algorithm will try to determine for each item to be classified which individual classifier(s) is best qualified to trigger the correct solution.

But not only is it important for the ensemble algorithm to choose the right candidate to classify each particular instance, it is also essential that the individual classifiers be as complementary as possible. Ideally, we want a set of classifiers that, given an oracle picking the best candidate every time, achieves a 100% accuracy on the classification task. Generally, an ensemble technique can be evaluated by looking at the complementary nature of the components it contains¹ and the degree to which it approaches oracle-type decision making.

Bagging (*“bootstrap aggregating”*) is an ensemble machine learning technique conceived by [Breiman 1996] which tries to create an ensemble of classifiers, by training them on different re-samplings of the same data set. These re-samplings are created by making bootstrap replicates of the training set and using these as data sets to train new classifiers. So typically, bagging does not involve using different learning algorithms to create a set of classifiers. It rather uses a single learning algorithm trained on different data sets, which are created by randomly selecting instances (with replacement) from a training set and placing them in a number of new training sets, that are equal in size to the original training set.

Even though the new training sets are consistent with the original data, each set will yield a different instance space, since the selection mechanism ensures that some instances from the original training set will not occur in the newly built training sets, while other instances will occur several times, creating a different distribution of instances from one set to the next. After the training sets are used to train new classifiers, each of them will propose a classification for the instances in the test set, after which a simple majority voting mechanism can be used as the final predictor. Whereas the original training set would have provided the classifier with a possibly unsurmountable systematic bias, bagging ensures that each predictor is based on a different resampling of the original training set, thereby introducing for each classifier a different bias towards the final classification.

The success of the bagging approach is largely dependent on whether the selection mechanism is able to generate new data sets that are representative of the original dataset, yet varied enough to yield complementary classifiers. [Breiman 1996] indeed points out that bagging is more suitable for resolving

¹ Although some methods such as stacked classifiers do not create their own components.

biases of unstable predictors, such as connectionist methods, rather than stable methods, such as nearest neighbor approaches.

Regardless of the classifier used, bagging does seem to render good performance on a number of NLP problems, including PP-attachment [Abney et al. 1999; McLauchlan 2001], part-of-speech tagging [Hoste and Daelemans 2000], document classification [Koehn 2002] and syntactic parsing [Henderson and Brill 2000; Collins 2000a].

6.1.1 Bagging Treebank Parsers

[Henderson and Brill 2000] describes bagging (and boosting (Section 6.2)) experiments with a treebank parser. This requires a slight adaptation of the bagging approach, the algorithm of which is reproduced here:

Given: A corpus (again as a function) $\mathcal{C} : S \times T \rightarrow N$, S is the set of possible sentences, and T is the set of trees, with size $m = |\mathcal{C}| = \sum_{s,t} \mathcal{C}(s,t)$ and parser induction algorithm g .

1. Draw k bootstrap replicates $\mathcal{C}_1 \dots \mathcal{C}_k$ of \mathcal{C} each containing m samples of (s,t) pairs randomly picked from the domain of \mathcal{C} according to the distribution $D(s,t) = \mathcal{C}(s,t)/|\mathcal{C}|$. Each bootstrap replicate is a bag of samples, where each sample in a bag is drawn randomly with replacement from the bag corresponding to \mathcal{C} .
2. Create parser $f_i \leftarrow g(\mathcal{C}_i)$ for each i
3. Given a novel sentence $s_{test} \in \mathcal{C}_{test}$, combine the collection of hypotheses $t_i - f_i(s_{test})$ using the unweighted constituent voting scheme of [Henderson and Brill 1999]

The parser mentioned in step (2) of the algorithm is a distribution of Collins' model 2 parser described in [Collins 1997]. The combination method

	LP	LR	$F_{\beta=1}$	Gain	Exact	Gain
Original Parser	88.7	88.5	88.6	NA	34.9	NA
Initial	88.4	88.3	88.4	0.0	33.3	0.0
TrainBestF (15)	89.5	88.8	89.2	0.8	34.6	1.3

Table 6.1: Results for the [Henderson and Brill 2000] bagging experiment

in Step (3) refers to earlier work [Henderson and Brill 1999] on combining several treebank parsers. The constituent voting scheme involves several parsers voting on the inclusion of constituents in the parse. If a majority of the parsers include a certain constituent in their parse, it is included in the parse proposed by the constituent voting. [Henderson and Brill 1999] also describes conditions under which *the constituent voting [...] combination technique[s] are guaranteed to produce sets of constituents with no crossing brackets*. Detailed information on these conditions and the method to combine these constituents into a full parse are however unfortunately lacking in [Henderson and Brill 1999]. Since we have already defined an efficient simple weighted voting mechanism for treebank parsers in Chapter 3 (p. 73), we will use this as a combination technique in the bagging experiments described in this chapter.

Results for the [Henderson and Brill 2000] bagging experiment (summarized in Table 6.1) show a clear gain in using a bagging approach. The original parser achieved a 88.6% F-score and a 34.9% Exact Match accuracy score, while the “Initial” system (consisting of only one bag) scored a 88.4% F-score. A system consisting of 15 bags yielded the best scores on the training set and was used to parse the test set: it achieved a 89.2% F-score and a 34.6% Exact Match Accuracy. It is unclear why [Henderson and Brill 2000] define gain in terms of the observed difference in accuracy with the Initial (1 bag) system and not the original parser. In any case, the bagging approach yields a .6% performance increase compared to baseline accuracy, but loses out on exact match accuracy.

6.1.2 Relation to GRAEL

Bagging a treebank parser involves creating a number of data sets that differ from each other in terms of the distribution of the grammatical units in the individual bags. Some structures will not be featured in some bags, while they may occur multiple times in another bag, because structures from the training set are distributed randomly over a fixed set of bags, but with replacement, meaning that the same structure may occur several times in different bags. And even though the bags differ from one another, bagging tries to make sure that the aggregate distribution of all bags approaches that of the original training set. But ultimately, the success of a bagging experiment depends on whether the bags complement one another well enough to make a difference over the original data set. This is largely dependent on the nature of the data itself, but also on the way in which the bags are compiled.

The GRAEL-1 method appears to be similar in concept to bagging: in GRAEL-1 a collection of tree-structures is randomly distributed over a group of agents, or “bags”. But in a bagging approach, the newly created data sets are roughly the same size as the original data set, thanks to the creation of replicates **with replacement**, whereas the number of tree-structures that agents in a GRAEL society hold, equals the number of trees in the original data set, divided by the number of agents in the society.

The agents replenish their grammars through an extended series of language games, which for a fairly limited period of time (defined as the state of beneficial confusion in Chapter 5 (p. 156)), produces a number of agents that have grammars that are fairly different from each other in terms of the distribution of grammatical structures contained. Disregarding the development of the agents over time and only considering the grammars in the society at that moment, one might think of the GRAEL-society as a collection of bags. The difference with the bagging approach however is twofold: whereas bagging tries to find a collection of data sets that, considered as a whole, approach the distribution of the original training set, the grammars in the GRAEL-society at that point were not optimized to mirror the original training set, but rather to perform a particular task, i.e. parsing. We have shown in Chapter 5 that these are not the same, as a parser using a grammar optimized by GRAEL-1 outperforms a parser trained on the original distribution of the training set. The 2nd difference with bagging is that there is

no majority voting in a GRAEL-society: typically, only one agent is picked to parse the test set. We will however describe an experiment in Section 6.3.3 in which agents in a society are “bagged”.

6.2 Boosting

Boosting is related to bagging as an ensemble learning algorithm in that it also uses a collection of classifiers that are typically powered by the same learning algorithm. We noted with respect to bagging that different classifiers are created that each introduce their own particular bias towards classification. There is however no internal feedback: each bag is given a set of sentences and does not “reflect” on its data set. Boosting introduces some form of internal feedback by keeping track of misclassified instances to consequently assign them more weight than correct instances in the subsequent resampling.

Adaptive Boosting *AdaBoost* [Freund and Shapire 1996]) is one of the most popular ensemble learning techniques in machine learning. Similarly to bagging, the AdaBoost algorithm generates several classifiers from a training set. In the initial phase, the weights for all instances in the different training sets are equal. When one of the classifiers makes a mistake, the weight for that particular instance is increased, forcing the classifier in the next round to focus on those examples.

This ensemble learning technique has been applied to an increasing number of machine learning tasks of natural language, among which text classification [Schapire and Singer 2000], text-filtering [Schapire et al. 1998], tagging [Hoste and Daelemans 2000; Abney et al. 1999], PP-attachment [Abney et al. 1999] and syntactic parsing [Henderson and Brill 2000]. Boosting generally appears to outperform bagging as an ensemble learning algorithm [Dietterich 2000], but other studies [Hoste and Daelemans 2000] note that even though boosting yields an important error rate reduction on exceptional cases, a higher error rate on regular cases can also be observed, because of overgeneralization of the exceptions.

6.2.1 Boosting Treebank Parsers

[Henderson and Brill 2000] redefines some of the AdaBoost methodology to make it applicable for treebank parsing. We reproduce the algorithm here:

Given: corpus \mathcal{C} with size $m = |\mathcal{C}| = \sum_{s,t} \mathcal{C}(s,t)$ and parser induction algorithm g . Initial uniform Distribution $D_{-1}(i) = 1/m$. Number of iterations, T . Counter $t = 1$.

1. Create \mathcal{C}_t by randomly choosing with replacement m samples from \mathcal{C} using distribution D_t .
2. Create parser $f_i \leftarrow g(\mathcal{C}_i)$ for each i
3. Choose $\alpha_t \in]0,1[$

4. Adjust and normalize the distribution. Z_t is a normalization coefficient. For all i , let parse tree $\Theta_i \leftarrow f_i(s_i)$. Let $\delta(\theta, c)$ be a function indicating that c is in parse tree θ , and $|\theta|$ is the number of constituents in tree θ . $T(s)$ is the set of constituents that are found in the reference or hypothesized annotation for s .

$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i) \sum_{c \in T(s_i)} (\alpha |1 - \alpha| |\delta(\theta'_i, c) - \delta(\theta_i, c)|)$$

5. Increment t . Quit if $t \geq T$
6. Repeat from step 1
7. The final hypothesis is computed by combining the individual constituents. Each parser ϵ_t in the ensemble gets a vote with weight α_t for the constituents they predict. Precisely those constituents with weight strictly larger than $\frac{1}{2} \sum_t \alpha_t$ are put into the final hypothesis.

	LP	LR	$F_{\beta=1}$	Gain	Exact	Gain
Original Parser	88.7	88.5	88.6	NA	34.9	NA
Initial	88.1	88.1	88.1	0.0	33.3	0.0
TrainBestF (15)	89.4	88.3	88.8	0.8	33.0	-0.3

Table 6.2: Results for the [Henderson and Brill 2000] boosting experiment

Constituents are considered correct if they are featured in the tree-structure proposed by the parser, as well as in the original reference structure from the training set. Points are added or deducted according to the number of correct constituents that are parsed. The value of α can be tweaked to express preference for a specific kind of accuracy measure.

The experimental results which [Henderson and Brill 2000] report on the boosting experiment can be found in Table 6.2. Again it compares the original parser to an initial system with only one resampling and a system with 15 resamplings, which had proved to yield the best results on the training set. The results show that bagging is to be preferred over boosting. The performance increase over the original parser is much lower for the F-score and there is a performance decrease for exact match accuracy. It seems that the boosting method used is not appropriate for this task, as even the initial system with only one resampling already shows a significant decrease on accuracies compared to the bagging counterpart.

[Henderson and Brill 2000] suggest that the underwhelming results for boosting are caused by a violation of the weak learning criterion. Their data analysis showed that 11.2% of the training corpus could simply not be learned by the parser². An extra experiment was conducted in which those particular sentences were trimmed from the corpus, but this yielded lower results. [Henderson and Brill 2000] hypothesize that the boosting algorithm itself performed better, but that useful lexical information was lacking that had previously been compiled from those sentences.

²This was tested by creating 39.832 parsers each trained on one sentence. 4.764 of those parsers could not parse their own sentence correctly.

6.2.2 Relation to GRAEL

Boosting as an ensemble learning method mirrors some of aspects of the GRAEL-1 approach. GRAEL-1 shared the bagging concept of distributing a group of tree-structures to provide a number of different resamplings of the original training corpus. Boosting further ties in with GRAEL-1 by effectively implementing a form of **error-driven learning**, providing a re-distribution of grammatical structures based on errors made by the parser. Whereas bagging tries to create an aggregate of bags that approaches the original training set distribution, boosting can optimize the distribution to achieve higher accuracy scores in parsing.

This is very similar to what is going on in GRAEL-1, albeit in a different way: if **agent1** parses a sentence wrong, **agent2** will cause **agent1** to *increase the weight* of the relevant (sub)structure in his grammar. GRAEL-1 therefore also clearly incorporates a form of error-driven learning. The difference with boosting lies in the fact that the agents start out with grammars that are totally distinct from one another, i.e. at the onset of a GRAEL-society, a tree-structure will only occur once in each agent's E-language. The language games consequently make sure that through a form of inter-agent error-driven learning, grammatical information is shared in a way that optimizes the distribution of probability mass for actual parsing, rather than to mirror the distribution of the original training set.

6.3 Experimental Setup and Results

Figure 6.1 describes the experimental setup. We are comparing the performance of three classifiers: the fittest agent in a GRAEL-1 society, the standard bagging or boosting approach and a classifier that interprets the agents in a GRAEL-society as bags. We will refer to the results in Chapter 5 for the GRAEL-experiments (except for the “bagging GRAEL-1” approach). The same training set - test set division was used for all data sets. We used two data sets for our experiments: the toy ATIS-corpus and the keystone WSJ-corpus. We used the PCFG+PMPG parser (Chapter 3) to parse the sentences.

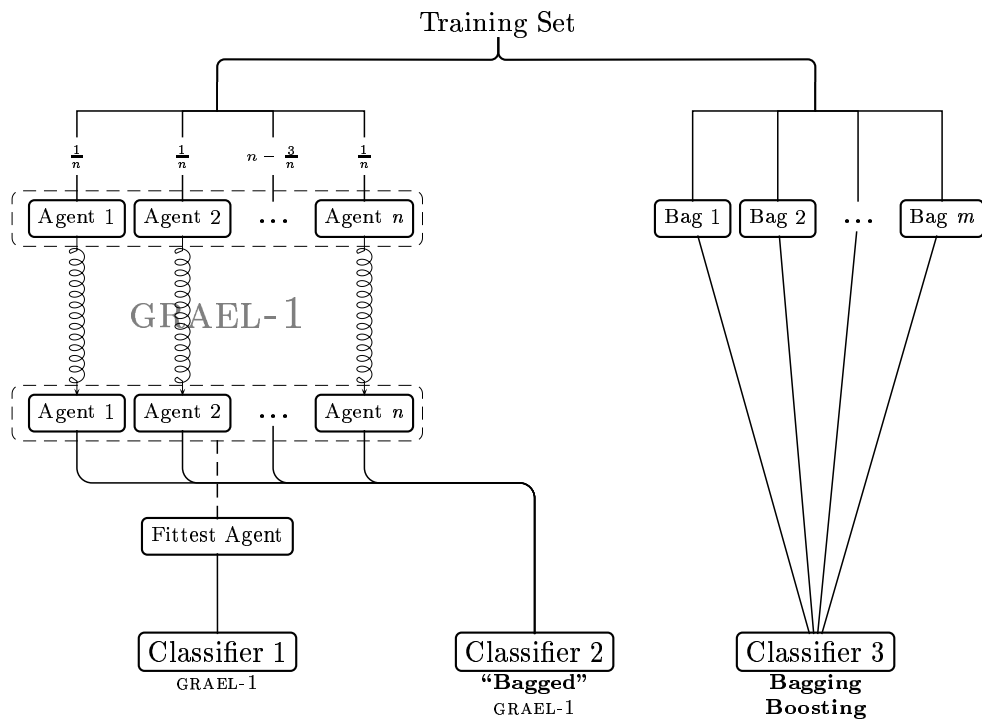


Figure 6.1: Comparing three classifiers: GRAEL-1, "Bagged" GRAEL-1 and Bagging/Boosting

	ATIS		WSJ	
	Exact Match	$F_{\beta=1}$ -score	Exact Match	$F_{\beta=1}$ -score
Baseline	70.7	89.3	16.0	80.5
GRAEL-1 (5)	72.4	90.9	—	—
GRAEL-1 (10)	77.6	92.1	—	—
GRAEL-1 (20)	77.6	92.1	—	—
GRAEL-1 (50)	75.9	92.2	22.2	80.7
GRAEL-1 (100)	75.9	92.0	22.8	81.1
1 Bag	67.2	80.1	13.3	78.6
10 Bags	72.4	91.7	17.1	81.0
15 Bags	75.9	91.8	20.0	81.4

Table 6.3: Baseline, GRAEL-1 and Bagging results

6.3.1 Bagging

Table 6.3 displays the exact match accuracy and F-scores for the baseline model, the standard PCFG+PMPG parser that was hitherto used. It also displays scores of the GRAEL-1 system, using sexual procreation, the combination of accuracy and understanding as fitness function and majority voting to halt the society³. We notice a significant gain for all GRAEL-1 models over the baseline model. Increasing population size over 20 agents seems to decrease Exact Match accuracy. Note however that there is only an absolute difference of one sentence between the 20 agent-society and the 50-agent society, which produces a seemingly high accuracy shift, due to the small size of the test set.

The first experiment implements a bagging approach to grammar optimization. The method for creating the bags is the same as in [Henderson and Brill 2000], but we employ a different method for combining the decisions made by the different bags. [Henderson and Brill 2000] uses a method called unweighted constituent voting (described in [Henderson and Brill 1999]) in which the individual classifiers can contribute constituents to the final decision. We will however use the weighted voting mechanism for full parsers

³Refer to Chapter 5 (Section 5.3.4) for details on important adjustments to the GRAEL-1 system made for the experiments on the WSJ.

described in Chapter 3 (p. 73) as a combination technique in the bagging experiments. This implements a simple weighted majority voting method as follows: the 10 most probable parses are gathered from the parse forests of the individual parsers. Next, their respective probabilities are added. This returns an ordered miniature parse forest of at most $10 \cdot n$ parses. The parse with the highest probability is the parse proposed by the majority voting method.

We have tested three different bag sizes: an initial system with one bag and two systems with 10 and 15 bags respectively. Due to time constraints, we have not dynamically determined the number of bags, unlike the [Henderson and Brill 2000] experiment. Our main focus is the system consisting of 15 bags, as this was reported by [Henderson and Brill 2000] to yield the best results on a training set and close to the best result on the test set in their experiments.

When we look at the results of the bagging experiment, we also notice a considerable increase when using a bagging approach on the ATIS-corpus. Using only 1-bag⁴, exact match accuracy is much lower for the ATIS-corpus. This can probably be attributed to the fact that ATIS, although fairly homogeneous, is a small treebank, so that the absence of certain key structures, may well deteriorate parsing accuracy on a test set. Using 10 bags on the ATIS-corpus, improves on the baseline model, while using 15 bags (comparable to [Henderson and Brill 2000]) further improves on the accuracy. The bagging approach is however not able to outperform the GRAEL-1-approach on the ATIS-corpus.

The situation is more or less reversed for the WSJ-corpus: similar to the experiments in [Henderson and Brill 2000], using only one bag does not degrade performance significantly over the base-line model. But the system consisting of 15 bags outperforms GRAEL-1 by a significant margin. Nevertheless, GRAEL-1 (100 agents) parses 69 more sentences completely correct than a 15-bag system does. The advances made by the bagging approach on the experiments described in this paper are comparable to those in [Henderson and Brill 2000], even though the overall scores are still lower.

To conclude, the results are somewhat of a mixed bag: the GRAEL-society

⁴Even though this would a priori seem to limit parse accuracies, we include this method for the sake of comparison with [Henderson and Brill 2000].

outperforms the bagging approach by a significant margin on the ATIS-corpus, yet the situation is reversed on the WSJ-corpus, even though the difference between the two methods is considerably smaller. GRAEL-1 seems to do a better job at improving exact match accuracies than bagging. This means that GRAEL-1 on average generates more incorrect constituents than the bagging approach does, but is better at finding a global solution for a sentence.

6.3.2 Boosting

[Henderson and Brill 2000] describe an interesting application of the AdaBoost algorithm on treebank parsing. Although it provides a very well-balanced definition of boosting on this domain, the results are such that it is not always clear what the actual effect of boosting proper is. Whereas the bagging experiment employed unweighted constituent voting [Henderson and Brill 1999] to decide on the ensemble's decision, Step 7 in the aforementioned boosting algorithm involves a weighted voting scheme. Even though the boosting algorithm provides such a weight on the fly, following [Henderson and Brill 1999], these weights must have also been implicitly present in the data of the bagging experiment. [Henderson and Brill 2000] also describes an equation to compute α (Step 3) which can be adapted to optimize the algorithm for precision, recall, or F-measure. This α is used to adjust the weight of correctly classified constituents.

We will employ a different boosting algorithm, one that is less intricate, but which should provide a clearer insight into the effects of boosting proper. The adjustments of weights in our boosting experiments is defined as follows: over the course of 10 iterations, adjust the weight w of a sentence i , by looking at its proposed parse P and the correct parse T as follows:

$\delta(t_i, t_j)$: a function that counts the number of constituents in tree-structure i that can also be found in tree-structure j

$$w_i = 1 - \frac{2 \cdot \frac{\delta(P,T)}{\delta(P,P)} \cdot \frac{\delta(P,T)}{\delta(T,T)}}{\frac{\delta(P,T)}{\delta(P,P)} + \frac{\delta(P,T)}{\delta(T,T)}}$$

This gives each sentence a weight that is roughly the inverse of its F-

	ATIS		WSJ	
	Exact Match	$F_{\beta=1}$ -score	Exact Match	$F_{\beta=1}$ -score
Baseline	70.7	89.3	16.0	80.5
GRAEL-1 (5)	72.4	90.9	—	—
GRAEL-1 (10)	77.6	92.1	—	—
GRAEL-1 (20)	77.6	92.1	—	—
GRAEL-1 (50)	75.9	92.2	22.2	80.7
GRAEL-1 (100)	75.9	92.0	22.8	81.1
1 Bag	67.2	80.1	13.3	78.6
10 Bags	72.4	91.7	17.1	81.0
15 Bags	75.9	91.8	20.0	81.4
Boosting (1)	67.2	80.9	13.7	77.5
Boosting (10)	74.1	91.7	17.6	80.8
Boosting (15)	75.9	91.7	21.0	81.0

Table 6.4: Baseline, GRAEL-1, Bagging and Boosting results

score. When inducing a grammar for these sentences in the next iterations, the probability of each constituent will be multiplied by this weight + 1, thereby redistributing the probability mass over the grammatical structures to try and resolve previous erroneous analyses.

After 10 iterations, we are left with a number of classifiers that each have a certain kind of specialty. Rather than giving the classifiers a weight in the final decision, as in [Henderson and Brill 2000], this weight is intrinsically provided by the majority voting mechanism previously described: a miniature ordered parse forest is culled from the n best parses in the parse forest of each classifier. A classifier that is sure of its decision will express this in the probability of the tree-structures it proposes, thereby supplying a bigger weight towards final classification in the combination method.

Table 6.4 describes the results of the boosting experiment compared to the GRAEL-1 experiments and the bagging experiment. Again, we notice a significant performance increase when using 10 or 15 bags over the baseline model. Boosting using only one bag significantly reduces accuracies for the ATIS, as well as the WSJ-corpus. The 10 to 15 bags systems achieve a similar result on the ATIS-corpus as bagging, i.e. lower than the GRAEL-1 approach.

GRAEL-1 (50)	ATIS		WSJ	
	Exact Match	$F_{\beta=1}$ -score	Exact Match	$F_{\beta=1}$ -score
1 “bag”	75.9	92.2	22.2	80.7
5 “bags”	77.6	92.2	22.9	81.1
10 “bags”	77.6	92.3	23.1	81.1
15 “bags”	77.6	92.3	23.2	81.1
50 “bags”	75.9	92.0	22.7	80.5

Table 6.5: “Bagging Agents” Results

For the WSJ-corpus, it is a close tie between boosting and GRAEL-1. GRAEL-1 does outperform boosting on exact match accuracy, but the F-scores are very close. Neither of them can outperform bagging on this account.

6.3.3 “*Bagging*” Agents

One final experiment tried to combine the sensibilities of bagging and GRAEL-1 by considering agents as bags and applying a majority voting method on their decision. The basis for this experiment was the 50-agent society: we vary the number of bags for this experiment as well, by considering the n fittest agents of a GRAEL-society as bags. The previously described weighted majority voting mechanism consequently compiles an ordered miniature parse forest and produces the most probable tree-structure.

Table 6.5 displays the results for this experiments. The first line (1 bag) expresses the standard GRAEL-1 accuracy, achieved by the single fittest agent in the society. The 5, 10 and 15 bags-system yields a slight performance increase on the accuracies of the ATIS-corpus. But for both datasets, a 50 bags-system, i.e. using all agents in the society, decreases performance.

The 5, 10 and 15-bags systems however yield some significant performance increases on the WSJ-dataset. Exact Match accuracy advances significantly and so does the F-score. Using majority voting, the 50-agent society is now able to climb up to a 100-agent society in terms of F-score. Interestingly, a side-experiment that used 10 “bags” from a 100-agent society yielded no performance increase over the 50-agent counterpart whatsoever.

	ATIS		WSJ	
	Exact Match	$F_{\beta=1}$ -score	Exact Match	$F_{\beta=1}$ -score
Baseline	70.7	89.3	16.0	80.5
GRAEL-1 (50)	75.9	92.2	22.2	80.7
Bagging (15)	75.9	91.8	20.0	81.4
Boosting (15)	75.9	91.7	21.0	81.0
GRAEL-1 (50-15)	77.6	92.3	23.2	81.1

Table 6.6: Comparison: Baseline - (bagged) GRAEL-1 - Bagging - Boosting

6.4 Concluding Remarks

Table 6.6 provides a comparison between the different systems discussed in this chapter. The results show clearly that all methods described here improve accuracy on both the ATIS and the WSJ dataset. GRAEL-1 outperforms bagging and boosting on all accounts in the ATIS-experiment. This can be explained by the small and homogeneous nature of the ATIS-corpus and may therefore corroborate the claim made in [Breiman 1996] that bagging is less suited for stable predictors, as well as the hypothesis put forward in [Hoste and Daelemans 2000] that bagging has the advantage over boosting for classification of typical instances.

Bagging outperforms GRAEL on the WSJ-corpus, but only on a constituent level. When we are interested in improving exact match accuracy, GRAEL appears to have the edge over bagging and boosting. We believe that this is due to the way in which grammatical knowledge in a GRAEL-society is shared, i.e. using substructures, which may provide a predilection for larger structures in parsing. Boosting is not able to outperform bagging, nor GRAEL-1 as a grammar optimization method.

The scores for GRAEL-1 seem closely tied to those of the boosting experiments. Apart from the F-score on the WSJ-corpus, GRAEL-1 outperforms boosting, but only by a small margin. The similarity in results may be explained by both GRAEL-1 and the focal point of boosting, i.e. error-driven learning. Boosting re-adjusted the probability mass of grammatical structures if an error was made on them. This is more or less what is going on in GRAEL-1 as well, but rather than enforcing the information locally, the

re-adjustment on the probability mass is being made in another grammar. The slight edge GRAEL-1 has over boosting may perhaps be explained by the fact that the non-local re-adjustment of grammatical information in GRAEL-1 applies more favorably on the task of parsing an **unseen** test set. Another possible explanation might be that boosting starts out with all the grammatical information present in the grammar, while agents in a GRAEL-1-society acquire new grammatical structures in language games, while at the same time, re-adjusting the probability mass to incorporate the new, as well as the old structures. The boosting approach would therefore be similar to a 5-agent society based on the ATIS-corpus, in which it was observed that the agents start out with too much grammatical information to warrant a flexible redistribution of grammatical structures.

Allowing for the fact that GRAEL-1 requires a lot of computational effort compared to bagging and boosting, it does seem to hold up very well to the established ensemble techniques as a grammar optimization technique. Also note that scores reported on the WSJ dataset were achieved on an approximation of GRAEL, whereas the ATIS experiments were conducted on a full instantiation of GRAEL. This may explain the reversed situation when moving from ATIS to WSJ.

The comparison of GRAEL and the ensemble learning techniques of bagging and boosting expose the similarities between the methods. Important differences are however apparent, indicating that GRAEL holds its own as an optimization method for corpus-induced parsing.

Scientist1: "Look at that! I never imagined a tree like that could even exist!"

Scientist2: "Normally it wouldn't, it must be some kind of mutation."

Scientist3: "That's right, probably caused by radioactivity."

Last Days of Planet Earth - ©1974

7

GRAEL-2 - An Agent-Based Evolutionary Computing Approach to Grammar Rule Discovery

In Chapter 5 we discussed a method for the optimization of treebank grammars. By distributing an annotated corpus of tree-structures over a group of agents and having them interact in a setting that mirrors the task at hand, the agents replenished their grammars with new information, while re-adjusting the probability mass over the grammatical structures in an optimal manner for parsing.

We compared this method in Chapter 6 to other ensemble techniques. GRAEL-1, as well as the bagging and boosting algorithms, held one important trait in common: at no point during processing, was new grammatical information introduced in the system (society) that was previously unavailable to the aggregate components (agents, bags). The grammar optimization techniques were therefore geared towards redistributing grammatical structures to make them suitable for parsing.

Bagging tried to alleviate problems of classifier bias, by resampling the original training set a number of times to create an ensemble of classifiers, each bringing to the combination their own particular bias. We described boosting as an error-driven learning algorithm, which re-adjusted the grammatical information based on the errors the individual classifiers made, so that an ensemble of classifiers was created, each bringing their own line of specialty to the mixture. GRAEL-1 resembles boosting in that it is also an error-driven learning process, in which particular chunks of grammatical information are enforced on the basis of errors, although the re-adjustment was done in the other agent's grammar, rather than locally.

In this chapter, we will take one extra step and allow the agents to introduce grammatical information in the society that was *previously unavailable* to the aggregate of agents. The creation of new grammatical structures is however bound to strict rules and ultimately they are nothing more than adaptations of existing grammatical structures, i.e. grammatical structures that have undergone some form of **mutation**.

This ties GRAEL-2 in with some related research efforts in the induction of grammars using an evolutionary computing approach. Little or no effort has been made to apply genetic algorithms on the optimization and induction of grammars for natural language. Most of the research so far has been based on artificially constructed languages [Dupont 1994; Huijsen 1993; Kammeyer and Belew 1996; Keller and Lutz 1997a; Keller and Lutz 1997b; Lankhorst 1994; Lucas 1993; Lucas 1994; Wyard 1989; Zhou and Grefenstette 1986; Losee 1995]¹, but most researchers claim their insights are also portable to actual natural language data.

[Smith and Witten 1996] use tree structures as their syntactic representation, with nodes labeled as either AND or OR, which can be interchanged during mutation. Fitness of an individual is measured by counting its grammar size and its ability to parse test strings. They found that "*recurring patterns helped to reinforce partial inferences, but intermediate states of the model may include incorrect generalizations that can only be eradicated by continued evolution*". [Wyard 1991] and [Blasband 1998] also apply Genetic Programming to induce and optimize grammars. Closely related to this line of work, is [Antonisse 1991] in which grammar-based crossover is imple-

¹References from Literary Survey on Genetic Algorithms for NLP [Kool 1999].

mented, a feature lacking in [Smith and Witten 1996]. Crossover occurs by randomly splitting a sentence into two sentence fragments and interchanging them with two fragments of another sentence.

[Losee 1995] describes the LUST system in which an information retrieval system is powered by genetically evolving grammars. The grammars in this system are optimized to contain rules that are specifically suited for the IR-task. Optimization in the LUST-system takes place by mutation: the right hand side of fit rules are altered by combining fragments from fit rules with the same category on the left hand side. But the description of the mutation process in [Losee 1995] more hints of a crossover operation than actual mutation.

GRAEL-2 tries to provide a viable alternative for these methods, by addressing their weaknesses: the linguistic analysis of the generated data in [Smith and Witten 1996] suffers from the fact that their linguistic representation (AND/OR) is too weak to offer any insight in the performance of their grammar for actual natural language, so that it cannot be objectively evaluated. The grammars described in [Blasband 1998; Losee 1995] seem foremost geared towards the practical application in which they need to feature (spoken dialog systems and information retrieval respectively) and are not able to provide any insights in the dynamics of grammar development in an evolutionary context itself, the way GRAEL is able to. The grammars used in [Wyrd 1991] are based on artificially constructed languages and it is not clear how they can be applied to large amounts of language data. [Antonisse 1991] provides a more interesting account of grammatical development in an evolutionary context, but its crossover operation is too relaxed, creating a large amount of uninterpretable structures². Also, the main goal in [Antonisse 1991] is to create a framework for the evolution of all types of grammars (linguistic and non-linguistic alike).

The advantages of GRAEL-2 compared to the aforementioned research can therefore be summarized as follows: it provides a corpus-based, therefore **portable** method for grammar optimization and induction that can handle **large amounts of natural language data**. The **modular** architecture allows a GRAEL-society to be tweaked to provide grammars for specific pur-

²Similar to the **Crossover Mutation** operation described in Section 7.1 and tested in Section 7.3.4.

poses, while the corpus-based approach provides an **objective** touchstone for the grammars that the system yields.

We will discuss the different aspects of mutation in Section 7.1, go into the experimental setup in Section 7.2 and discuss the results on the ATIS and WSJ corpus in Sections 7.3 and 7.4 respectively. Finally, we conclude by summarizing the insights this chapter has granted us in Section 7.5.

7.1 Mutation

Chapter 4 defined three ways to perform mutation on an agent's grammar:

1. Crossover Mutation: this kind of crossover only occurs when random crossover is enabled. Chapter 5 described a limited set of experiments in which agents were allowed to randomly crossover substructures in tree-structures, but with only one restriction: the structures that were crossed over had to carry the same node label. Crossover mutation abandons this restriction, meaning that crossover operations such as the one described in Figure 7.1 are permitted. From this example one may gather that this operation will yield more ungrammatical structures than grammatical ones. It is only included in this chapter because it resembles the approach proposed by [Antonisse 1991].

2. Internal Mutation: this is a different type of mutation altogether. This kind of mutation does not occur by attaching constituents to different nodes, in the way crossover mutation does, but adapts grammatical structures, by adding and/or deleting nodes from them and changing node labels (see Figure 7.2). Internal mutation occurs when an agent mutates the grammatical structures in his own grammar. This can in principle occur at any time, but we apply this kind of mutation only during procreation, since we have the *noisy channel mutation* operation (cf. *infra*) to create mutated structures during the normal course of the society. Internal mutation occurs when an agent is born into the GRAEL-society with grammatical structures inherited from its ancestor(s). To some of those structures a number of mutation operations are applied. This relates GRAEL-2 to the general genetic algorithm technique of creating new instances, by crossing over and mutating existing instances.

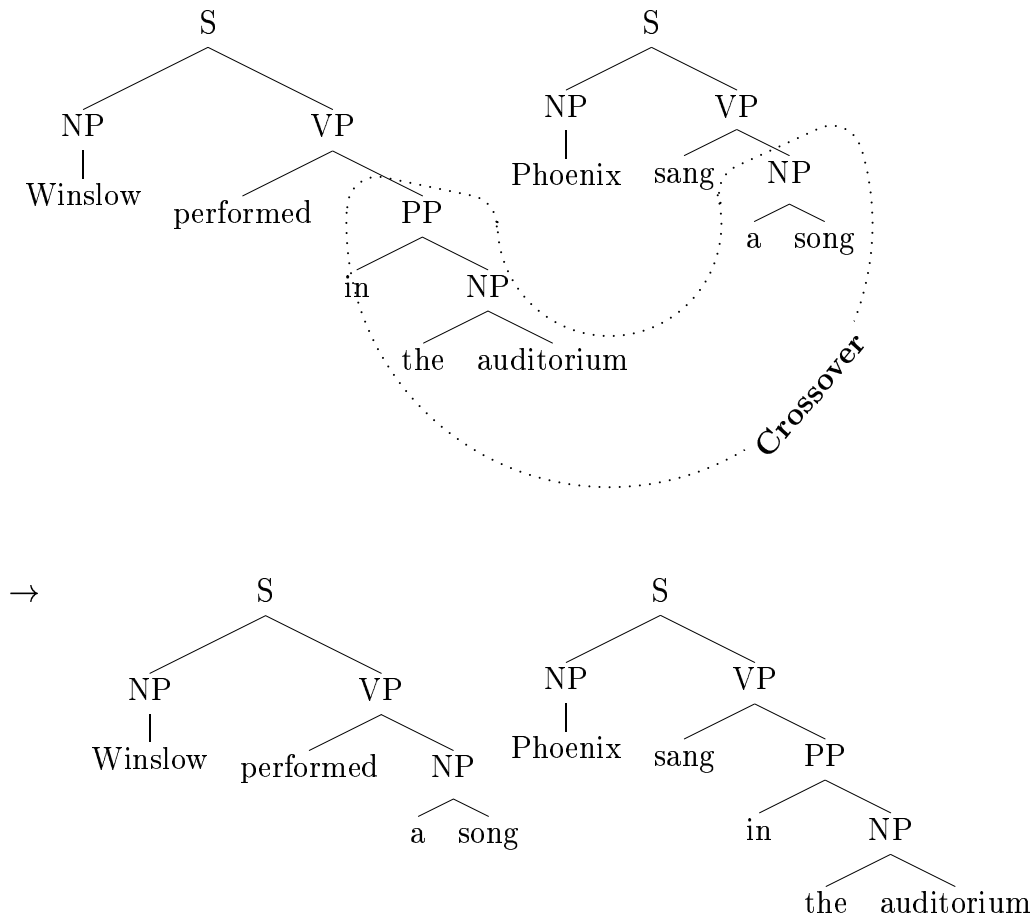


Figure 7.1: Example of Crossover Mutation

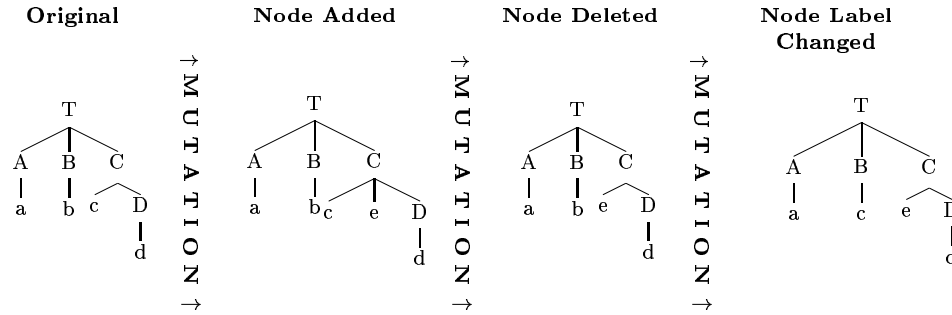


Figure 7.2: 3 different (Micro-)Mutation Operations

3. Noisy Channel Mutation: this mutation occurs in the context of language games and provides a way to more rapidly cause new grammatical information to be created than *internal mutation* is able to do. Noisy channel mutation occurs when **Agent1** suggests a minimal correct substructure to **Agent2** during a language game, but is hindered by a virtual noisy channel, which may cause **Agent2** to misunderstand the information sent by **Agent1**. This kind of mutation may involve adding nodes, deleting nodes or changing node labels (See Figure 7.2).

Let us look at the mutation operations exemplified in Figure 7.2. We have defined three different types: adding nodes, deleting nodes and changing node labels. We start off with a T-structure heading a structure with terminals *abcd*. The first type of mutation involves adding a terminal node so that the string turns into *abcde*. To accommodate the node another branch is added to a node. The location of attachment is limited to the node heading one of the neighboring terminal nodes. In our example, this means *e* can be attached to node **C** or **D**. The choice is made randomly. In the example in Figure 7.2 it is attached to the **C**-node.

The next mutation operation involves deleting a terminal. In the example, the deletion yields the terminal string *abed*. A deletion of a terminal results in the deletion of the superordinate node, unless there are other terminals headed by that node, as is the case in the example. Finally, terminals can also be changed: in Figure 7.2 this results in the terminal string *aced*.

The mutation operations so far described, mostly affected the terminal nodes, but one can also apply mutation to the grammatical structure on a higher level. This is exemplified in Figure 7.3. Adding a node then means

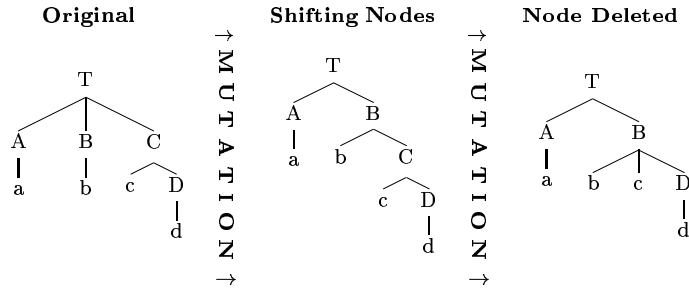


Figure 7.3: 2 Types of Macro-Mutation

that a node is deleted from its current position and adjoined to a different node, as in the example of Figure 7.3. Deleting a node involves shifting the entire subordinate structure one level up in the tree-structure. We do not consider changing category node-labels, nor adding new category nodes in the structure as macro-mutations.

We define two categories of mutation: micro-mutation, pertaining to mutations on terminals, and macro-mutation, which alters the higher-level structural properties, i.e. the attachment of category node-labels. When combined, micro-mutation always precedes macro-mutation, but there is no reason to do it the other way around.

7.2 Experimental Setup

We described a couple of alternative approaches to adding mutation to the GRAEL-system. We now turn to some experiments to see what kind of advantage the different possibilities can provide. As in Chapter 5, our main batch of experiments was conducted on the ATIS-corpus, with the WSJ-corpus providing corroborating results on a larger-scale data set.

We have suggested three different situations for mutation: **crossover** mutation, **internal** mutation and **noisy channel** mutation. Crossover mutation will be dealt with as an extra experiment in Section 7.3.4, as little or no added insight is to be expected from those experiments and is only included for the sake of comparison with [Antonisse 1991]. The two other situations will be experimented on in isolation (denoted as im and ncm experiments

	Adding Nodes	Deleting Nodes	Changing Label
ADC	.3	.3	.3
A	1	0	0
D	0	1	0
C	0	0	1
Adc	.5	.3	.2

Table 7.1: 5 Calibration Settings for mutation

infra) and in combination (NCIM).

All GRAEL-2 systems will feature micro-mutation, as most of the *grammar sparseness* problems can be mostly located in structures directly covering terminals. But we will also test the combination with macro-mutation, to see if coverage can be gained by mutating higher-level structures as well.

We also vary some of the micro-mutation operations to test their effect on the performance of a GRAEL-2 society. Table 7.1 displays the combinations we will consider. The first default setting (ADC) applies equal weight to all three operations, meaning that the three types of mutation will occur an equal amount of times. The next three settings just look at the three different kinds of mutation in isolation. The results of these experiments allow us to apply an optimized weighting in the last calibration setting.

So far, we have not addressed the problem of evaluating GRAEL-2. Investigating different settings for GRAEL-2, we are mostly interested in their ability to generate structures that were needed for parsing, but were previously unavailable. But with only 58 sentences in a typical ATIS test set and a fairly limited number of constituents, it would be hard to identify significant differences using the corpus division used in GRAEL-1. To find a suitable test set for the GRAEL-2 experiment, we looked at the number of sentences in the ATIS-corpus that had at least one constituent³ not found in any other sentence. We counted 124 unique constituents in the ATIS corpus, i.e. rules that were only featured in one sentence (see Appendix F for an overview of these sentences). Those 124 rules were found in 97 sentences (16.8% of all

³“Constituent” is defined here as a single-level constituent, i.e. as a rewrite-rule.

sentences). This effectively means that an exact match structure can never be provided for these sentences, even by a parser trained on all the other tree-structures in the ATIS-corpus.

We took those 97 sentences and turned them into the test set. Even though this makes for the most unfavorable training set - test set partition imaginable (by definition yielding a 0% exact match accuracy score) it does provide a test set on which any favorable effect of mutation can easily be observed. A more fundamental problem to this approach however, is that it does not comply with the blind-testing requirements, since the test set is compiled by exploiting knowledge about the entire data set. But since the research issue here is to find a good method for covering marginal structures, the violation of the blind-testing principle is justifiable to some extent. And from a more narrow point of view, there is still blind-testing going, as the agents in GRAEL-2 do not exploit knowledge from the test set when optimizing their grammars, so that testing is still being performed on unseen data.

However, this does beg the question: how can agents in a GRAEL-2-society learn to parse marginal structures, when all that is left in the society are non-marginal structures to practice on. After the 97 aforementioned sentences are separated from the rest of the training set, there are about 20 sentences in the rest of the data set left that feature a unique constituent, as they shared that constituent with one or more of those 97 sentences that now constitute the test set. This means that the GRAEL-2-society does contain a limited number of rules that are unique to a particular sentence.

The nature of the language games in GRAEL however would cause those unique structures to be quickly distributed over the society, so that after a while, inter-agent communication resembles using a training set to parse the training set itself. The structures that previously were unique, will become common-place through inter-agent knowledge sharing. This however poses a serious threat to the effectiveness of mutated structures, as the agents will have no real use for them. The original structures will almost always be preferable, so that the mutated structures are never used and are therefore treated as noise. The first experiment will show that it is the strict distinction between E-language and I-language that renders the mutation operation useless in our first instantiation of GRAEL-2 and subsequent experiments will therefore relax the distinction.

One final adjustment we need to make relates to determining the age of an agent. In a noisy channel mutation model, new grammatical information is added all the time, so that the default definition of end-of-life does not apply here. As in the GRAEL-1 WSJ-experiment we implement the lifespan of an agent, using the following simple method:

$$\begin{array}{c|c} \text{First Generation} & \text{Consecutive Generations} \\ \hline n * 2 + \text{rand}(n) & n + \text{rand}(n) \end{array}$$

n = the number of agents in the society

The first generation of agents is allowed to build up its grammar for a longer period of time to establish a firm grammatical basis for future generations. Consecutive generations occur randomly with a minimal interval of n runs, with n being defined as the number of agents in a society.

The following table displays an overview of all the experiments conducted on GRAEL-2, minus the extra-curricular experiments described in Section 7.3.4:

(1)	ATIS	20 agents (I \leftrightarrow E)	micro	NCM	ADC	
(2)		20 agents (I \leftrightarrow E)	micro	NCM	ADC	
(3)					ADC (SE)	
(4)					ADC (Sp)	
(5)					A	
(6)					D	
(7)					C	
(8)					Adc	
(9)					NIM	Adc
(10)					NCIM	Adc
(11)		+macro			NCM	Adc
(12)		10 agents (I \leftrightarrow E)	+macro	NCM	Adc	
(13)		50 agents (I \leftrightarrow E)	+macro	NCM	Adc	
(14)	wsJf	100 agents (I \leftrightarrow E)	+macro	NCM	Adc	

We conducted a total of 14 experiments with the GRAEL-2 system. The first experiment (1) maintained the strict division between I-language and E-language, which will prove not to be tenable. It used noisy channel mutation

(NCM) and an equal distribution of mutation operations (ADC) in a GRAEL-2-society in which new agents were created through sexual procreation. The 2nd experiment (2) adds interaction between I-language and E-language and provides the first workable instantiation of GRAEL-2. Experiments (3) and (4) investigate single epoch and splicing instantiations of the GRAEL-2 system. We then look at the different calibrations of mutation operations in experiments (5) to (8) (cf. Table 7.1). Internal mutation and its combination with noisy channel mutation is investigated in experiments (9) and (10), after which experiment (11) adds macro-mutation operations to GRAEL-2. The best combination of settings for the 20-agent ATIS society was consequently used for different society sizes and data sets in experiments (12) to (14)⁴

7.3 Experiments: ATIS

We start off with the first batch of experiments on the ATIS-corpus. We already noted that the GRAEL-2 experiments differ from the GRAEL-1 experiments in the test set being used. GRAEL-1-societies were evaluated on a randomly compiled 58-sentence test set. Since GRAEL-2 is mainly geared towards supplementing grammars with possibly useful grammar rules, rather than optimize grammars to achieve maximum performance on a held-out test set, we argued in Section 7.2 for the offbeat evaluation method of creating a worst-case scenario test set of 97 trees, of which at least one constituent is unknown to the training set. To compensate for the sizable test set, we do away with any GRAEL-processes that involve a validation set, so that we are left with enough critical mass to constitute a good GRAEL-society. Experiments have shown that the **Understanding** fitness function by itself can achieve top-of-the-line agents, while looking at a plateau in understanding accuracy provides a halting point that usually halts the society at a favorable time. We are left with a 481 sentence training set (to be distributed over the GRAEL society) and a 97 sentence test set.

Table 7.2 displays the baseline accuracy achieved by a PCFG+PMPG parser trained on the 481 sentences of the training set and tested on the

⁴For reasons of time, we assumed that the best setting for the mutation operations was independent of population size and data set.

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0
GRAEL-1	624	712	86.3	64.5	73.8	0/97	0

Table 7.2: Baseline and GRAEL-1 Accuracies on 97-sentence test set

97 sentences of the test set. The Exact Match Accuracy score is by definition 0%, since all of these sentences contain at least one rule that is unknown to the training set. With 25 sentences for which no parse could be generated, the number of constituents created by the parser is limited to 687⁵, as opposed to 968 constituents in the annotated corpus. With 578 constituents correct, this still produces a reasonable precision score of 84.1%, but the recall score suffers. As a consequence, the overall F-score is very low at **69.8%**. Also included is a standard GRAEL-1 system which yields some performance increase, but an underwhelming F-score.

7.3.1 20 agents

The first experiment with GRAEL-2 involved noisy channel mutation, in which structures were mutated while being transferred from one agent to the next. Preliminary experiments (also described in Section 7.3.4) showed that assigning a 50% chance of a noisy channel mutation occurring during the transfer of a grammatical structure, helps to create a considerable amount of new structures in an appropriate amount of time, without overpowering the original grammatical content to any great extent.

The experiment featured an equal distribution between the three different micro-mutation operations: there is 33.3% chance that a terminal node is added, 33.3% chance that a terminal node is deleted and 33.3% chance that a terminal node label is changed. Note that this distribution is not exclusive and that there is for example an 11.1% chance of two mutations happening and a 3.0% chance of all three happening.

⁵Note that we do not allow the parser to propose partial parses, like agents are allowed to in language games.

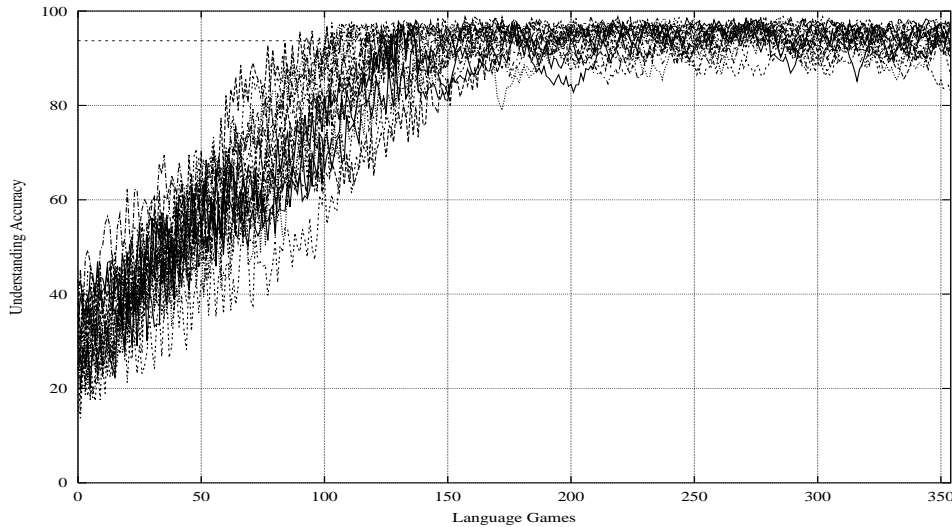


Figure 7.4: GRAEL-2 - 20 Agents - NCM - ADC - No interaction I-language/E-language

Figure 7.4 shows the course of this experiment expressed in the inter-agent understanding accuracy. This graph looks very similar to the GRAEL-1 counterpart (Chapter 5, p. 141), except for some agents that seem to disperse the plots somewhat: some agents seem to benefit from the mutated information in the early stages of the society (runs 60-100), while other agents are harmed, but overall, the addition of new grammatical information does not change the overall course of the experiment too much, compared to GRAEL-1. Since we have no validation set, we do not include the fittest agent plot for this experiment as we did for the GRAEL-1 experiment.

As mentioned before, we halt the society when understanding accuracies observed in language games are leveling. Using the understanding accuracy halting procedure (halting point at run 191) yields an F-score of **69.1%** and an exact match accuracy of 0% when we use the grammar of the fittest agent. Table 7.2 shows that the grammar induced from all the agents in the GRAEL-2 society does indeed perform better on the labelled precision score: the wealth of grammatical information in the entire society causes more constituents to be parsed. But whereas recall rises, the system loses out on precision. Apparently many more grammatical structures are created, but many of them are also wrong. Furthermore, there are still no sentences

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0
GRAEL-1	624	712	86.3	64.5	73.8	0/97	0
GRAEL-2 (1)	602	774	77.8	62.2	69.1	0/97	0
GRAEL-2 (all)	612	803	76.2	63.2	69.1	0/97	0

Table 7.3: Baseline, GRAEL-1 and GRAEL-2 Accuracies on 97-sentence test set

that are parsed completely correct, even though the number of unparseable sentences has dropped from 25 to 10 in GRAEL-2 (using all agents).

Adding Interaction

The disappointing results can be explained by the lack of feedback agents in the current GRAEL-2 society receive on the newly created structures. Agents in a GRAEL-2-society create new grammatical information, but clearly, there will be a preference for the “correct” structures over mutated structures that do not conform to the structures in the agents’ E-language. As a consequence, mutated rules are hardly ever used in language games, so that the agents never receive feedback in any way on the validity of the newly created structures. This means that all mutated structures carry the same low probability and are basically considered as noise in the grammar.

Also, in a generation-based GRAEL-society, low-probability rules tend to disappear from the society over time, effectively rendering the entire mutation operation useless. In a single epoch society, this would perhaps not pose a big problem, since all grammatical information is retained throughout its life-span. When the need arises, these marginal structures may still be called upon for parsing. But this still leaves us without useful probabilities for these structures for parsing difficult unseen data. Furthermore, the mutation operations in GRAEL-2 generate a large amount of new grammatical structures, some of which are useful, but most are not. Clearly this begs for a generation-based approach to distinguish useful grammatical information from ineffective structures over time.

So we are left looking for a method to provide feedback on the validity of mutated structures in such a way that they are treated as actual possibly useful grammatical constructs, rather than noise. We argued for the validity of the strict separation between I-language and E-language in Chapter 5. An experiment in which interaction was allowed between these two components, proved that it has a negative effect on the performance of a GRAEL-1 society, even though it did not hinder the society in its development. GRAEL-1 was geared towards redistributing probability mass in agents' grammars to optimize the construction of tree-structures that need to be as close to the original as possible. The goal of GRAEL-2 is however different: it does not explicitly try to create grammars to achieve the best F-score on a test set, but rather tries to experiment with a large amount of possibly useful structures and distinguish them from bad structures in an evolutionary agent-based setting. The negative effect that removing the barrier between I-language and E-language had on GRAEL-1 need therefore not be relevant for GRAEL-2.

In fact, if we allow each agent to parse his own sentences with his I-language, in which newly acquired, mutated structures reside, we actually find a way to include these structures in the language games. Some of these structures will indeed be featured in the updated tree-structures that constitute the E-language, so that in subsequent language games, the minimal correct substructures that are transferred contain previously mutated information. To nudge the agents to use newly acquired tree-structures in their E-language, we double each newly acquired (and therefore possibly mutated) structure in the agent's I-language during the first generation. This will provide them with sufficient probability mass to overtake the resident structures if necessary.

Figure 7.5 displays the course of this experiment. The first 10 language game runs return stable results, but as soon as the mutated structures gain power the understanding accuracies are very dispersed. There is a substratum of agents that yields similar understanding accuracies until around the 120th language game run. The society at this point is already sprouting 4th generation agents and the diversity in the agents is such that there is almost a 50% gap between the F-score of the most understanding and least understanding agent.

Almost surprisingly, after 200 language game runs the society settles down and seems to converge. The mutated information of the initial generations

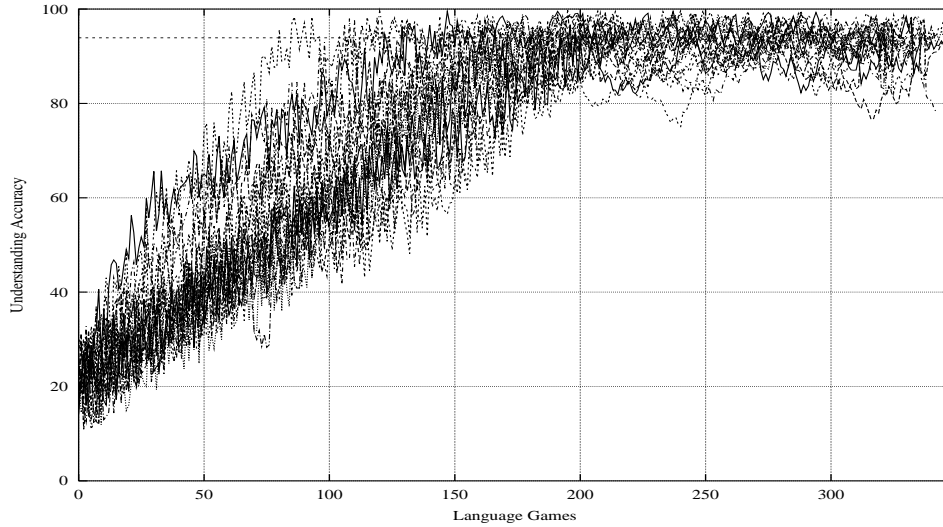


Figure 7.5: GRAEL-2 - 20 Agents - NCM - ADC -I-language—E-language

has become commonly accepted grammatical information and the agents' E-languages at this point remain largely unaffected by any mutated structures that are acquired beyond this point. The convergence phase differs from the convergence found in the equivalent GRAEL-1 society (Chapter 5, p. 141) in the variance of the understanding accuracies, but the overall course of the experiment is remarkably similar.

The results displayed in Table 7.4 are very encouraging. The GRAEL-2 society that introduces interaction between I-language and E-language (GRAEL-2 I) outperforms the original GRAEL-2 society (GRAEL-2 NI) on every account. The fittest agent's grammar, as well as the grammar compiled from all agents produces more constituents, of which many are correct. This increases recall scores by more than 15%. Whereas precision suffered from the negative side-effects apparent in the GRAEL-2-society without interaction between I-language and E-language (GRAEL-2 NI), it is now up to the same level of baseline accuracy. The overall F-score jumps to **79.4%**, an increase of almost 10%. The fittest agent's grammar parses two sentences completely correct (2 sentences are still unparseable), while the entire society adds an extra sentence to that count (with 0 sentences unparseable).

The interaction between I-language and E-language is clearly beneficial

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0
GRAEL-1	624	712	86.3	64.5	73.8	0/97	0
GRAEL-2 NI (1)	602	774	77.8	62.2	69.1	0/97	0
GRAEL-2 NI (all)	612	803	76.2	63.2	69.1	0/97	0
GRAEL-2 I (1)	732	875	83.7	75.6	79.4	2/97	2.1
GRAEL-2 I (all)	747	910	82.1	77.2	79.6	3/97	3.1

Table 7.4: Baseline, GRAEL-1 and GRAEL-2 Accuracies on 97-sentence test set

for GRAEL-2. It has proved to be a good method for assigning mutated structures some ballpark probabilistic value based on inter-agent communication. The new GRAEL-2 system is able to generate more constituents. This means that also more correct constituents are generated. But compared to the baseline model, GRAEL-2 I loses out on precision. So even though the mutated structures allow the parser to generate many more constructions that were previously not available, there does seem to be a lot of noise in the grammars as well, causing them to lack precision.

Varying generation methods

The next experiments will try and look underneath the hood of GRAEL-2 to see what experimental parameters may perhaps be limiting its precision. One of the possible causes may be that sexual procreation does not agree with GRAEL-2 the way it does with GRAEL-1. We therefore look at the alternative methods of splicing and the single epoch society. The latter is of particular interest to us, as no grammatical information is discarded at any time: agents keep stacking up grammatical information without restriction.

Figure 7.6 shows the plot for the single epoch experiment. Overall, understanding accuracy are more dispersed over the graph, even in the convergence phase compared to the GRAEL-1 counterpart (Chapter 5, p.121). Some very high F-scores are being obtained in inter-agent communication, but even in the convergence phase, F-scores down to 80% are not an exception.

We would also like to draw attention to some peculiar behavior apparent in the plots in Figure 7.6. Starting at the 50th language game run, one particular agent seems to have acquired some very beneficial collection of grammatical structures, as its understanding accuracy on other agents scores. Due to the constantly changing E-languages in the society, understanding accuracies vary to a great extent from one run to the next, but this agent maintains its status, even after living on as his own offspring, until around the 120th run, when the rest of the society catches up with him. There is also another agent around the 70th run that breaks loose from the society. The data shows that it has engaged in a couple of language games with the aforementioned agent in a relatively short period of time. The “educated” agent benefits from the grammatical structures provided by the strong agent and starts a series of successful language games of its own. On the other side of the scale, we notice an agent that engages in a long series of unsuccessful language games starting from the 40th run, and only catching up to the other agents around the 100th run, after it dies off and its slot is occupied by another agent. The initial mutated structures it has acquired provided it with a particularly inadequate I-language. The data shows in fact that all the other agents achieve their worst understanding scores trying to parse this particular agent’s E-language as well. Eventually though, convergence does occur and the agents’ grammatical systems are similar to one another.

The results of the single-epoch experiment (GRAEL-2 (SE)) compares favorably to the crossover-experiment (Table 7.5). The fittest agent’s grammar is unable to parse two sentences in the test set, while there are no unparseable sentences for the grammar induced from the entire society. This means that more constituents are generated and also more correct constituents, which has a positive effect on recall. But precision loses out compared to the crossover society. This is of course a direct consequence of the single epoch method: no grammatical information is filtered from the society, which means that we ultimately have a huge coverage of grammatical structures. But a larger set of rules, many of them constituting nothing more than noise, implies that a large portion of the probability mass is lost to useless structures. This seems to negatively affect the precision score.

Table 7.5 also displays the results for the GRAEL-2 society using asexual procreation (GRAEL-2 (Sp)). Not much is to be learned from this experiment. The results are significantly worse than those of the crossover-based GRAEL-2

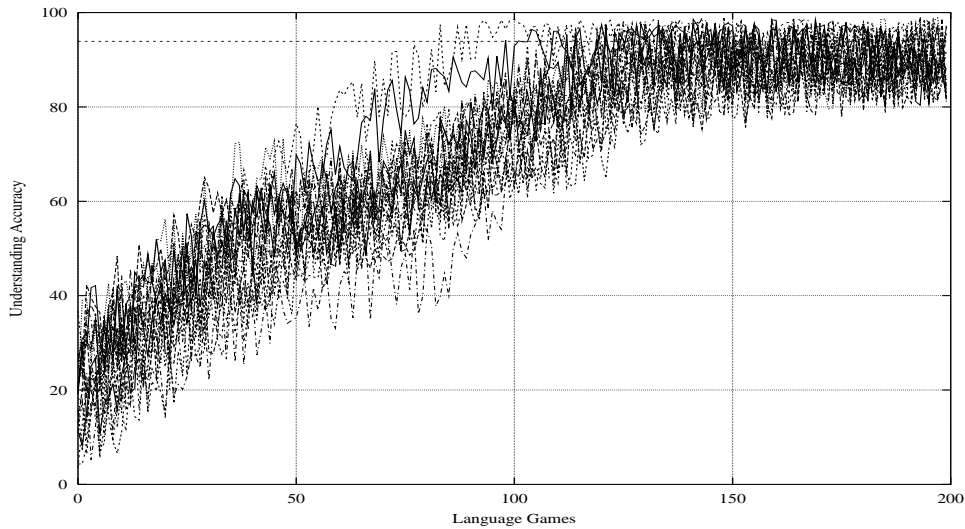


Figure 7.6: GRAEL-2 - 20 Agents - NCM - ADC - I-language—E-language - **Single Epoch**

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2 NI (1)	602	774	77.8	62.2	69.1	0/97	0.0
GRAEL-2 (XO)(1)	732	875	83.7	75.6	79.4	2/97	2.1
GRAEL-2 (SE)(1)	757	920	82.3	78.2	80.2	4/97	4.1
GRAEL-2 (SE)(all)	760	931	81.6	78.5	80.0	4/97	4.1
GRAEL-2 (Sp)(1)	688	835	82.4	71.1	76.3	2/97	2.1
GRAEL-2 (Sp)(all)	696	851	81.8	71.9	76.5	2/97	2.1

Table 7.5: Baseline and GRAEL-2 Accuracies on 97-sentence test set

society. This can be attributed to the fact that agents are born with a smaller and less cleverly compiled grammar than in a crossover-based society. They are therefore more susceptible to be disturbed by newly acquired, and often noisy grammatical information. Exact Match accuracy is the same though, and precision is higher than in a single-epoch society, indicating that it was able to filter out some noise, but not to the same extent as the crossover-based society was able to do.

Altering the generation-method in a GRAEL-2 society does not seem to resolve the precision issues. We will now take a look at the mutation operations in isolation to see what type of mutation yields the best coverage on marginal structures.

Altering Mutation Operations

First, we look at the mutation operation that adds nodes in isolation. This will give us a rough idea of how useful it is to create new grammatical information by adding elements to existing constituents. This type of mutation would seem to alleviate problems we identified in Chapter 3 with respect to flat NP-structures, for example in a constituent like *restriction code AP/57*. This is represented in the ATIS-corpus as:

NP → NN NN sym sym sym CD CD

Being a highly specific constituent, this kind of rule is unlikely to be induced from the training set. But given a constituent that more frequently occurs in the ATIS-corpus, like *flight number L 4 0*, represented by the following rule:

NP → NN NN sym CD CD

a series of addition mutation operations might eventually yield the rule required to parse *restriction code AP/57*.

Table 7.6 describes the results of using the mutation operator that adds node to constituents in isolation (GRAEL-2 (A)), compared to the default experiment using all three operations, interaction between I-language and

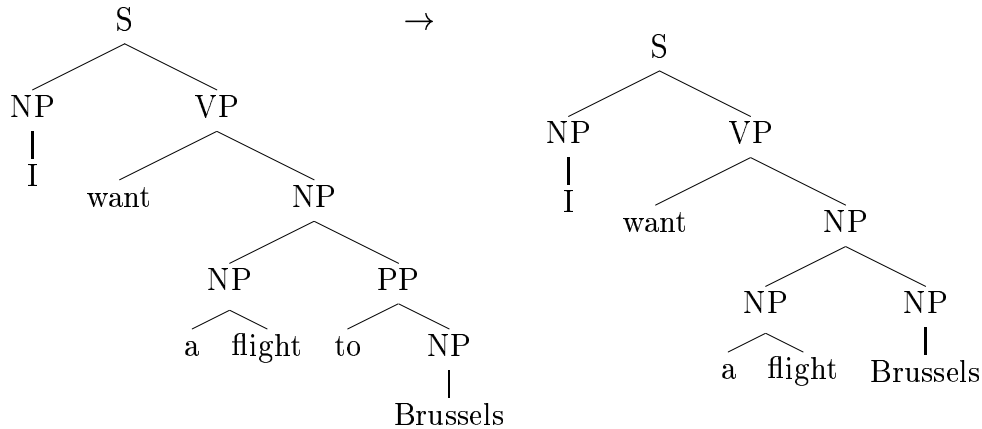


Figure 7.7: Mutation by Deletion

E-language and a crossover-based GRAEL-2-society⁶. The results are quite interesting: GRAEL-2 produces many more constituents than the baseline model, proving that it is able to trigger analyses that were unavailable to the training set. But compared to the default GRAEL-2 experiment, significantly fewer constituents are created. This is a direct consequence of the mutation operation that adds nodes to constituents: constituents contain on average a larger amount of terminals, so that flatter structures, consisting of fewer constituents can be generated. This has a beneficial effect on the precision scores, which is finally in an acceptable league. But the reduced number of constituents however negatively affects the recall score and the F-score.

It is also interesting to note that exact match accuracy benefits from the isolated mutation operation: 4 sentences are parsed completely correct. Two of those sentences had not been parsed correctly by any other previously discussed GRAEL-2 algorithm before.

The mutation operation that deletes nodes from constituents for very short NPS like the 2nd one in the sentence *There were 3*. This requires the rewrite rule:

$$\text{NP} \rightarrow \text{CD}$$

This kind of rule can easily be created by deleting a terminal in a con-

⁶The understanding accuracy plot for the experiment provides little or no added information, as it runs a similar course to the default (ADC)-experiment.

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2 (ADC)	732	875	83.7	75.6	79.4	2/97	2.1
GRAEL-2 (A)(1)	707	822	86.0	73.0	79.0	3/97	3.1
GRAEL-2 (A)(all)	714	843	84.7	73.8	78.9	4/97	4.1
GRAEL-2 (D)(1)	715	953	75.0	73.9	74.4	1/97	1.1
GRAEL-2 (D)(all)	721	970	74.3	74.5	74.4	1/97	1.1
GRAEL-2 (C)(1)	669	720	92.9	69.1	79.3	0/97	0.0
GRAEL-2 (C)(all)	681	735	92.7	70.4	80.0	0/97	0.0

Table 7.6: Baseline and GRAEL-2 Accuracies on 97-sentence test set: Adding Nodes

stituent like *2 friends*(NP \rightarrow CD NN), which is also featured in the ATIS-corpus. Deleting terminals in constituents however, can also have an effect on a higher level, as exemplified in Figure 7.7. In this structure the terminal node *to* is deleted. Unless there are other terminal nodes on the same level, the superordinate node is deleted as well and the subordinate structure(s) attaches to the first non-terminal node it encounters. This creates a new structure: NP \rightarrow NP NP. This example shows that the deletion operation should not be considered as an operation that only affects structures containing terminal nodes, even though that is its main focus.

The results of using deletion in isolation provide an interesting illustration of the effects of this mutation operation (GRAEL-2 (D)). Deletion eventually creates a whole set of rules containing only one terminal on the right-hand side. This means that a parse can possibly be construed by using a single category for each terminal in the sentence and building up the superordinate structure on top. This yields parses consisting of many constituents as is evident from the results. Many constituents are generated of which a surprisingly high percentage of correct structures, considering the careless parsing behavior these results belie. Compared to other GRAEL-2 instantiations however, the results are very low.

On inspection of the data however, we notice that the deletion mutation operation is not without merit. There is a strong distinction in parsing

behavior between this society and the society that used a mutation operation that only adds nodes with many structures found by either method not found by the other. This may seem trivial, but it does indicate that there is quite a large performance increase to be gained from combining the two.

Table 7.6 also shows results for the mutation operation that just changes terminal nodes. This can be thought of as a consecutive deletion and addition within the same constituent at the same location. Changing a node is structurally the least radical measure, as it does not involve changing the number of branches in a grammatical structure. It moreover serves as a fast way to diversify structures containing terminal nodes, like NPs.

The results in Table 7.6 show that this mutation operator only generates a limited number of constituents compared to the other operations. This is due to the fact that (a) it does not reduce the average number of terminals per node, in the way that the deletion operation did, and (b) it does not seem to produce low-level constituents that allow good higher-level structures to be built on top of it in the way the adding operation seemed to do. Exact Match accuracy for this mutation operation is 0%. Further corroborating the claim that this mutation operation does not yield good structures globally is the fact that 10 sentences could not be parsed. All the more surprising however then that precision is so high: this type of mutation may not yield a lot of good constituents, and it may not incorporate them in a full tree-structure very well, but what it does do well, it does very well. Data analysis shows that changing terminal node labels is a very precise way to mutate structures. Most of the time the mutated structure is not relevant, but when it is, it is quickly attributed a probability that elevates it to the status of an average corpus-induced rule.

A **comparative quantitative data analysis** of the results of the experiments described in Table 7.6 will allow us to fine-tune a weighted combination of the three mutation operations. Table 7.7 shows that mutation operation A finds 88 constituents that none of the other mutations is able to find. It finds 21 constituents that mutation operation D finds, but not C, etc.

This table reveals some interesting facts: the A mutation is a very useful operation: it achieves the best F-scores of all mutations in isolation, and is able to generate quite a lot of constituents the other mutations can not.

Correct by All: 579 constituents

Constituents				Sentences			
	A	D	C		A	D	C
A	88	21	19	A	2	1	0
D	21	93	22	D	1	0	0
C	19	22	49	C	0	0	0

Table 7.7: Comparative Quantitative data analysis - Mutation operations

The deletion operation (D) is an interesting case: its lacking parsing skills are apparent from the F-scores and the data analysis alike, but on the other hand, it is able to generate 93 constituents none of the other mutation operations can. The weighted combination should therefore include the D-type sensibilities. There is in principle no grammatical structure that C can create through mutation that a combination of A and D can not create either. Even though its straightforward operation limits the amount of constituents that its GRAEL-2-society creates, it does achieve high precision scores, and is still able to be the only system to find 49 particular constituents. Comparative quantitative data analysis on a sentence level is futile, because of the extremely low exact match accuracy scores reported.

This data analysis provides a hopeful prospect for a fine-tuned weighted combination of the three mutation operations. If we (wrongfully) assume that the inclusion of one mutation operation does not affect the performance of the other, an oracle that can choose the best constituent each time, would yield a recall score of 90.0% (871/968). However, this is assuming that the mutation operations act independently, while the experiment with the unweighted combination in the GRAEL-2 (ADC)-society suggests otherwise. Also, it assumes that constituents created as an effect of the different mutation operations would somehow all fit perfectly in one parse for each sentence, which does not hold true either. On the other hand, the combination of mutation operations might yield a beneficial side-effect not evident in any of the societies using the operations in isolation. In any case, we can reasonably expect the most optimal weighted combination to achieve a score that is significantly lower than 90.0%, but it is a good upper threshold to keep in mind.

Based on the data in Tables 7.6 and 7.7 we propose the following weight distribution:

	A	D	C
Adc	.5	.3	.2

This provides a good balance between the best performing mutation operation (A), the careless, but beneficial deletion mutation (D) and the straightforward C-operation. It is hoped that the combination can find a good balance between D's and C's predilections for tree-structures with respectively many and few constituents and A's solid parsing behavior.

The results of the weighted balance are quite encouraging (Table 7.8). Although nowhere near the 90.0% upper threshold we suggested, the recall score of 80% this system (GRAEL-2(Adc)) achieves, is significantly better than any of the GRAEL-2 societies tested so far. The overall F-score is reasonable and precision, although not much higher than that of the baseline model, seems respectable. There are quite a number of constituents being parsed, most likely grace to the D-mutation operation, but overall precision is not sacrificed because of it. A pleasant surprise was exact match accuracy. All sentences featured in the right table in 7.7 are also parsed correctly by the weighted combination method, while the combination itself adds more sentences to the exact match accuracy score.

Macro Mutations

Also related to the mutation operations is the question whether or not macro-mutations have a beneficial (if any) effect on performance. So far we have dealt with micro-mutations, that exclusively mutate structures containing terminal nodes. We have seen that at least for the deletion mutation operation, this does sometimes entail structural changes on a higher structural level, but its effect is rather limited.

Macro-mutations like the ones exemplified in Figure 7.3 mutate higher level structures. Adding this type of mutation to GRAEL-2 provides some interesting results (Table 7.8). The exact match accuracy shows that macro-mutation enables two sentences to be parsed correctly that were previously

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2 (ADC)	732	875	83.7	75.6	79.4	2/97	2.1
GRAEL-2 (A)(1)	707	822	86.0	73.0	79.0	3/97	3.1
GRAEL-2 (D)(1)	715	953	75.0	73.9	74.4	1/97	1.1
GRAEL-2 (C)(1)	669	720	92.9	69.1	79.3	0/97	0.0
GRAEL-2 (Adc)(1)	771	902	85.5	79.6	82.5	5/97	5.2
GRAEL-2 (Adc)(all)	779	912	85.4	80.5	82.9	5/97	5.2
GRAEL-2 (+ macro)(1)	771	890	86.6	79.6	83.0	7/97	7.2
GRAEL-2 (+ macro)(all)	780	901	86.6	80.6	83.5	7/97	7.2

Table 7.8: Baseline and GRAEL-2 Accuracies on 97-sentence test set: Weighted combination of mutation operations

not parsable, but it does also allow for very flat structures to be created (most notably because of the node shifting operation), which negatively affects the number of correct constituents generated by the parser.

Although the +macro society shares many correct constituents with the default GRAEL-2 society that only performs micro-mutations, the addition of macro-mutation has a very beneficial effect on precision, as well as recall. Some very good higher-level structures are being created. And even though the macro-mutations generate a lot of grammatical noise, the dynamics of the GRAEL-2-society seem better able at weeding out macro-mutation noise than it is at discerning micro-mutation noise. The data-analysis shows that, though most of the *grammar sparseness* can be located at the NP-level, i.e. typically the lower regions of the tree-structure, many of the aforementioned 124 unique rules relate to higher-level structures as well (see Appendix F). It is clear that the addition of macro-mutation is beneficial to the performance of a GRAEL-2-society. Data analysis showed that there is about an equal amount of useful structures created by either macro-mutation, i.e. node-shifting and node-deletion. All subsequent experiments will therefore feature macro-mutation.

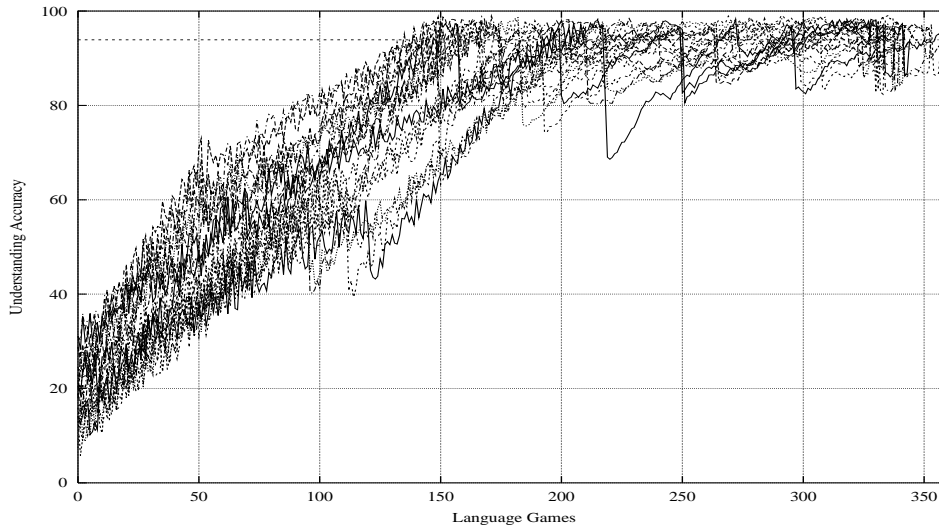


Figure 7.8: GRAEL-2 - 20 Agents - IM - Adc - I-language—E-language

Internal Mutation

The last two experiments before we investigate the dynamics of population size, deal with the situation in which mutation occurs. So far, we have only implemented noisy channel mutation. But we have also defined internal mutation, i.e. mutation that occurs at the time of conception of a newborn agent. A new agent is created by crossing over grammatical knowledge from two ancestors (identical to GRAEL-1). But before the agent starts processing the information, a portion of the structures in the I-language undergo a series of mutations. The newborn agent then provides the sentences in the E-language with tree-structures on the basis of his (mutated) I-language before entering the society.

Only applying internal mutation drastically cuts back the number of mutated structures that are going around. It also provides a more stable communication model and a society that is more stable overall. The question remains whether it provides sufficient variation to make a big impact on the worst-case test set we have compiled.

We performed a 20-agent experiment using internal mutation, a crossover-based society and the weighted combination Adc for mutation operations.

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2(ncm)	771	890	86.6	79.6	83.0	7/97	7.2
GRAEL-2(im)(1)	610	786	77.6	63.0	69.6	2/97	2.1
GRAEL-2(im)(all)	621	799	77.7	64.2	70.3	2/97	2.1
GRAEL-2(ncim)(1)	765	909	84.2	79.0	81.5	5/97	5.2
GRAEL-2(ncim)(all)	782	923	84.7	80.8	82.7	6/97	6.2

Table 7.9: Baseline and GRAEL-2 Accuracies on 97-sentence test set: Different Mutation situations

The course of the experiment is displayed in Figure 7.8: it provides a nice image of the course of action in this GRAEL-2-society. The society starts off processing similarly to a typical GRAEL-1-society. The information shared between agents is clean and does not undergo mutations. Around the 50th run, the first newborn agents are created. The effect on the data plots is not visible, due to the dispersed plots at this point. In later stages it is however clearly visible how some agent's understanding accuracy is signified by an extreme drop, caused by internal mutation. In the end, though, convergence does occur similarly to previous GRAEL-2 societies.

The result of this experiment can be read from Table 7.9 (results for GRAEL-2(im)). Internal mutation does increase performance over the baseline model, but it cannot outperform an NCM-type society. It does create a reasonable amount of constituents but is seriously lacking in the ratio of correct constituents. Internal mutation seems to create lots of new rules that create analyses for sentences that were previously unparsable, but the analyses on the whole are not very accurate. Perhaps internal mutation could perform better if it were allowed to run for many generations, but it appears unlikely that it would outperform an NCM-society.

A final experiment, the results of which can also be read from Table 7.9, combined internal mutation and noisy channel mutation. The combination introduces more mutated structures in the society, but not much in the way of added performance. More constituents are on average created, but at a lower precision rate, which is probably due to the larger amount of noise that is present in this society. Whereas in an NCM-type society, new generations

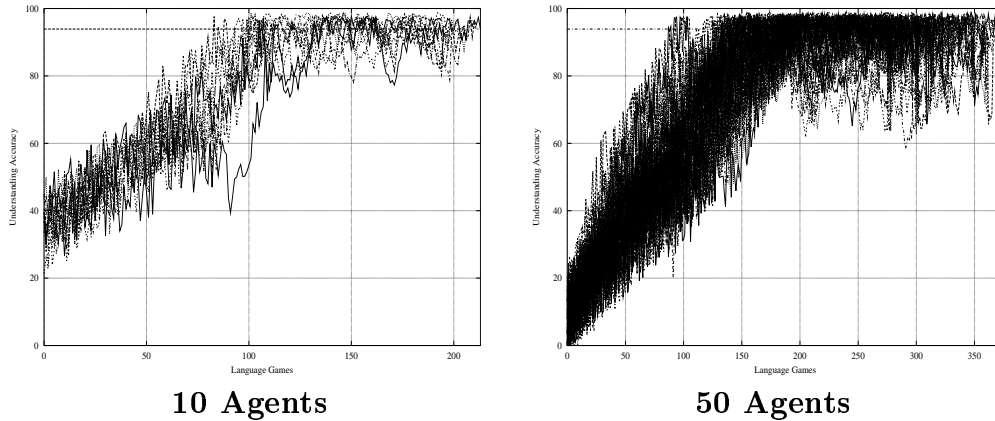


Figure 7.9: GRAEL-2 - 10 agents vs 50 agents

try to weed out noise from the society, a new generation in a NCIM-society typically introduces more. One slight edge this approach has over the other societies is a 6.2% exact match accuracy. There does not seem to be any other reason to abandon NCM as the mutation situation of choice however. It will therefore be used in subsequent experiments that look at different society sizes.

7.3.2 10, 50 agents

Having optimized the experimental settings for GRAEL-2 on a 20-agent society, we now apply the method on respectively a smaller and larger population size. For reasons of time, we therefore assume that the setting pertaining to the different mutation operations and the like are not affected by society size. The experimental results show that this assumption may not be as harmful as it seems, since results are very close to the 20-agent GRAEL-2-society and the data-analysis indicates that society has little or no effect on its performance.

The plots in Figure 7.9 show that the 10-agent GRAEL-2-society starts off with a coherent set of runs, from the 70th till the 100th run, there seems to be a lot of confusion with a range of over 50% between the best understanding accuracy score and the worst, indicating that the mutated information seems to be “settling in”. The society eventually converges not long after this period of confusion ends. The 50-agent society follows a typical GRAEL-

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2 (20ag)(1)	771	890	86.6	79.6	83.0	7/97	7.2
GRAEL-2 (10ag)(1)	761	888	85.7	78.6	82.0	4/97	4.1
GRAEL-2 (10ag)(all)	779	903	86.3	80.5	83.3	4/97	4.1
GRAEL-2 (50ag)(1)	777	923	84.2	80.3	82.2	7/97	7.2
GRAEL-2 (50ag)(all)	786	945	83.2	81.2	82.2	7/97	7.2

Table 7.10: Baseline and GRAEL-2 Accuracies on 97-sentence test set: Different Society Sizes

2-course, with the only difference that understanding accuracy scores in the convergence phase can drop as low as 80%, almost always because of newborn agents trying to adapt to the GRAEL-2-society.

Around the 100th language game run, we have a peculiar situation in which three agents break away from the society in terms of understanding accuracy. On inspection of the data, we noticed that an anomaly in the random selection of agents for language games, had caused these three agents to develop a miniature GRAEL-2-society, only playing language games with one another. This causes them to have unusually high understanding accuracies. Around the 110th run, this miniature society breaks up and rejoins the rest of the society.

Table 7.10 shows the results for the different society sizes. Each society size displayed in this table used a crossover-based GRAEL-2 society, employing noisy channel mutation, and the *Adc*-weighted mutation operation. The 10-agent society produces less constituents than the 20-agent counterpart, but at a higher precision rate, while recall is slightly lower. The 10-agent society seems to produce a less diverse selection of mutated structures, but its application of these structures is more meticulous. Overall, the F-score of the 10-agent GRAEL-2-society is within range of that of the 20-agent society. Note also that exact match accuracy is slightly lower, with the same 4 sentences parsed correctly by both the 20-agent and the 10-agent society, although the 20-agent society adds one extra sentence to its exact match score.

The 50-agent society presents a different situation. In this society, more agents provide a more diversified range of E-languages, containing a limited set of sentences. This seems to speed up the transformation of mutated structures into resident grammatical structures. As a consequence, there are more constituents generated, but on the whole not many more correct ones than in a 20-agent GRAEL-2-society. Precision is therefore trailing compared to the other societies, while the recall score is slightly higher. The overall F-score is the same. The exact match score however benefits from the more diverse range of mutated structures with up to seven sentences parsed correctly by a grammar compiled from all agents in the society.

To conclude, there is little to choose between the different society sizes, certainly in terms of F-score. While the 10-agent society has the edge on the precision front, the 50-agent society is able to squeeze out a few more marks on the exact match accuracy score. Overall, the 20-agent society seems to provide a workable middle-ground between the two society sizes.

7.3.3 Summary of Results and Discussion

Table 7.11 displays the full results of the GRAEL-2 experiments on the ATIS-corpus. We also include Table 7.12 which reports GRAEL-2 results on the ATIS-corpus, but for grammars that are compiled from all agents in the society, as opposed to just the fittest agent in Table 7.11.

The results show that a combination of noisy channel modeling and Adc can obtain a reasonable F-score of more than 80%. The main factor in increasing precision seems to be limiting the scope of the deletion operation even though it needs to be included in the weighted combination to allow for a good recall score. Comparing Table 7.11 with 7.12, we see that a grammar compiled from all agents in a GRAEL-society always performs better than that of the fittest agent. This is in direct contrast with the GRAEL-1 experiments in which the fittest agent almost always outperformed the full society, even on weaker fitness functions.

This can easily be explained by looking at the task at hand: GRAEL-1 tried to find the agent with the grammar containing the best tuned distribution of probability mass over a number of constituents. Agents differed from each other, not in the structures they held in their grammar, but in the way

Ag	I \leftrightarrow E	Gen	+Mac	M?	A.D.C	LP	LR	F $_{\beta=1}$	ExMa
Baseline						84.1	59.7	69.8	0.0
20	-	XO	GRAEL-1			86.3	64.5	73.8	0.0
20	-	XO	-	ncm	ADC	77.8	62.2	69.1	0.0
20	+	XO	-	ncm	ADC	83.7	75.6	79.4	2.1
20	+	SE	-	ncm	ADC	82.3	78.2	80.2	4.1
20	+	SP	-	ncm	ADC	82.4	71.1	76.3	2.1
20	+	XO	-	ncm	A	86.0	73.0	79.0	3.1
20	+	XO	-	ncm	D	75.0	73.9	74.4	1.1
20	+	XO	-	ncm	C	92.9	69.1	79.3	0.0
20	+	XO	-	ncm	Adc	85.5	79.6	82.5	5.2
20	+	XO	+	ncm	Adc	86.6	79.6	83.0	7.2
20	+	XO	+	im	Adc	77.6	63.0	69.6	2.1
20	+	XO	+	ncim	Adc	84.2	79.0	81.5	5.2
10	+	XO	+	ncm	Adc	85.7	78.6	82.0	4.1
50	+	XO	+	ncm	Adc	84.2	80.3	82.2	7.2

Ag: Population Size
 I \leftrightarrow E: Interaction Between I-language and E-language
 Gen: Generation Method
 +Mac: Macro Mutation allowed
 M?: NCM or IM or NCIM
 A.D.C: Adding Nodes, Deleting Nodes or Changing Node Labels
 XO: Crossover (Sexual Procreation)
 SE: Single Epoch
 Sp: Splicing (Asexual Procreation)

Table 7.11: GRAEL-2 ATIS: Complete Results Table (fittest agent only)

Ag	I \leftrightarrow E	Gen	+Mac	M?	A.D.C	LP	LR	$F_{\beta=1}$	ExMa
Baseline						84.1	59.7	69.8	0.0
20	-	XO	GRAEL-1			86.3	64.5	73.8	0.0
20	-	XO	-	ncm	ADC	76.2	63.2	69.1	0.0
20	+	XO	-	ncm	ADC	82.1	77.2	79.6	3.1
20	+	SE	-	ncm	ADC	81.6	78.5	80.0	4.1
20	+	SP	-	ncm	ADC	81.8	71.9	76.5	2.1
20	+	XO	-	ncm	A	84.7	73.8	78.9	4.1
20	+	XO	-	ncm	D	74.3	74.5	74.4	1.1
20	+	XO	-	ncm	C	92.7	70.4	80.0	0.0
20	+	XO	-	ncm	Adc	85.4	80.5	82.9	5.2
20	+	XO	+	ncm	Adc	86.6	80.6	83.5	7.2
20	+	XO	+	im	Adc	77.7	64.2	70.3	2.1
20	+	XO	+	ncim	Adc	84.7	80.8	82.7	6.2
10	+	XO	+	ncm	Adc	86.3	80.5	83.3	4.1
50	+	XO	+	ncm	Adc	83.2	81.2	82.2	7.2

Ag: Population Size
 I \leftrightarrow E: Interaction Between I-language and E-language
 Gen: Generation Method
 +Mac: Macro Mutation allowed
 M?: NCM or IM or NCIM
 A.D.C: Adding Nodes, Deleting Nodes or Changing Node Labels
 XO: Crossover (Sexual Procreation)
 SE: Single Epoch
 Sp: Splicing (Asexual Procreation)

Table 7.12: GRAEL-2 ATIS: Complete Results Table (full society)

the probability mass was distributed over them. In GRAEL-2 agents do differ from one another in terms of the grammatical information they hold. And since the task at hand is to parse difficult, marginal structures, bigger does actually mean better in the context of GRAEL-2.

But the data should also put things in perspective. The fittest agent's grammar in a generation-based society is considerably smaller than the grammar induced from all agents. The fittest agent's grammar is only 10% to 40% the size of the full society's grammar and this significant reduction (which can also be observed on parsing times) does not translate into an extreme performance drop. If we want our grammar to have the largest sweep in terms of recall, we can use the mogul-solution that inducing a grammar from the entire society provides. A more meticulous method that yields a more computationally attractive grammar is the standard GRAEL-method of inducing grammar rules from the fittest agent in the society provides.

We have argued for the case of our unconventional test set by stating that it allows us to measure more accurately the success of the mutated structures created by different instantiations of the GRAEL-2 society. Let us now look at some more detailed figures on the parsing behavior of GRAEL-2 compared to the baseline model.

There are a total of 97 sentences in the test set consisting of 968 constituents. There are 124 single-level constituents (i.e. rewrite rules) in the test set that are not found in the training set (see Appendix F). The **baseline** model retrieves 578 constituents of those constituents, but it is not able to find a parse for 25 out of 97 sentences. These 25 sentences consisted of 263 constituents, so that the baseline model is limited in the LP/LR-scores it can achieve, since it cannot cater to 263 constituents (only a relatively small percentage of which causes the sentence to be unparseable).

We can therefore redefine the score of the baseline model by looking at scores achieved on parsable sentences alone: the baseline model is then able to score 84.1% LP (578/687) and 82.0% LR (578/705) which results in an F-score of **83.0%**. This is however still significantly lower than the F-score reported in Chapter 3 (90.6%), indicating that the test set we compiled is indeed a very hard one to parse well.

GRAEL-1 performs better with an F-score of **73.8%**. Its optimization seems to have beneficial effects on parsing, even on a very tough test set. But

GRAEL-1 is also unable to parse the same set of 25 sentences that defeated the baseline model. The scores on the 72 parsable sentences are as follows: an LP-score of 86.3% (624/723), an increased LR-score of 88.5% (624/705) resulting in a **87.4%** F-score. But this is still much lower than the F-scores reported in Chapter 5 and also lower than what the baseline model achieved on a randomly compiled test set (Chapter 3).

The GRAEL-2-society using `Adc` with noisy channel mutation in a 20-agent crossover-based society that uses the fittest agent's grammar (henceforth GRAEL-2-BEST) is not marred by unparseable sentences at all. For each of the 97 sentences it is able to come up with at least one possible analysis. This means however that its F-score of **82.5%**, which basically constitutes the F-score on parsable sentences, negatively compares to the baseline model and GRAEL-1 F-scores on parsable sentences.

It is indeed important not to overestimate GRAEL-2-BEST simply by looking at the 10% increase in F-score. A grammar holding all possible rewrite rules would be able to parse all sentences as well and might therefore perform better than the baseline model on a global level, but it does not make for a good grammar. To really evaluate GRAEL-2 as a grammar induction module, we need to look at its success vis-a-vis the 124 rules that were not available to the training set and its success on those sentences that the baseline model were able to parse, but not correctly.

Let us first look at the performance of GRAEL-2-BEST on the list of unique rules. Table 7.13 provides an overview of how many of the 124 rules were found in the GRAEL-2-BEST grammar and how many actually showed up in the parses. The results are encouraging: more than 70% of the rules were retrieved by GRAEL-2-BEST⁷, while more than 50% were actually used in parses.

The NP-category, which account for almost half of the unique rules, is also the category that GRAEL-2 is able to discover best. But also higher-level structures, such as VPs do not perform bad either, thanks to the beneficial effect of macro-mutations. GRAEL-2 was unable to find useful mutations for the NX-constituents, which is due to the fact that only one sentence in the training set used this category. There were several instances of rules that

⁷Note that this not necessarily mean we have a good grammar induction system, as a mindless system generating all possible rewrite rules, would yield a 100% retrieval.

Total		In GRAEL-2 Grammar	In GRAEL-2 Parses
124	ALL	91	64
56	NP	53	39
18	VP	12	7
9	PP	7	5
8	FRAG	5	3
5	SQ	2	1
5	WHNP	2	1
5	X	3	3
4	ADJP	3	2
3	NX	0	0
3	QP	1	1
3	S	1	0
2	SBAR	1	1
1	ADVP	0	0
1	PRT	0	0
1	SBARQ	1	1

Table 7.13: GRAEL-2-BEST performance on list of unique rules

	705 constituents		263 constituents	
	#	$F\beta = 1\%$	#	$F\beta = 1\%$
Baseline	578	82.0	0	0
GRAEL-1	624	88.5	0	0
GRAEL-2-BEST	560	79.4	211	80.2

Table 7.14: Baseline vs GRAEL-1 vs GRAEL-2-BEST F-scores

were not discovered by GRAEL-2 because of their peculiar nature: one such example is $PP \rightarrow TO\ INTJ$. We would probably need a more radical mutation operation to discover this rule, but this would unavoidably make the task of distinguishing useful grammatical information from noise even harder. But despite the fact that some rules are still not covered by GRAEL-2, Table 7.13 does prove that GRAEL-2 is a workable method for grammar rule discovery.

We had identified 25 sentences that could not be parsed by the baseline model, nor by GRAEL-1. The mutated structures of GRAEL-2 however generate parses for all 25 sentences. Five out of the seven sentences that GRAEL-2-BEST is able to parse correctly are among these sentences. Furthermore, all five of these sentences featured at least one constituent out of the 64 constituents from the first line in Table 7.13. While 20 out of 25 sentences were parsable by other mutated rules, five of them seemed to require one of the highly specific structure in the list of 124 to trigger the correct parse.

Table 7.14 shows recall scores for two parts of the test set: the parsable sentences (containing 705 constituents) and the unparsable sentences (containing 263 constituents). GRAEL-2-BEST achieves a significantly lower recall score on the parsable sentences compared to GRAEL-1 and to a lesser extent the baseline model. But the recall score on the unparsable sentences shows that at least GRAEL-2-BEST is consistent and that it does not seem to make a distinction between “difficult” and “easy” sentences.

It is clear that GRAEL-2 provides a decent method for discovering peculiar grammatical constructions. But the results show that it is in fact very different from GRAEL-1: while the added structures in the grammar allow for a broader coverage, they also seem to clutter the grammar. This is evident from the comparison with GRAEL-1 on parsable sentences: GRAEL-1 was

able to optimize the probability mass in such a way as to provide a significant performance boost over the baseline model. But the added structures in GRAEL-2 seem to render this optimization more difficult, which brings the overall recall score back to the level of the baseline model.

7.3.4 Extra Experiments

In this section, we describe some extra experiments with GRAEL-2 that provide some extra insights and/or serve as a sanity check to the previously described experiments.

Crossover Mutation

One situation for mutation we still need to experiment on is crossover mutation. This occurs when random crossover is enabled in a GRAEL-society and agents can randomly crossover structures. The restriction that crossover can only occur for structures carrying the same node label is abandoned, which results in mutated structures (Figure 7.1).

We experiment on a GRAEL-2-BEST-society to which we added random crossover and crossover mutation. As is to be expected, the results in Table 7.15 prove that this type of crossover is indeed useless. Scores for this type of GRAEL-2-society (GRAEL-2-BEST (xover)) decrease all around. Data analysis shows that crossover mutation has no positive effect on a grammar whatsoever. This corroborates the results of [Antonisse 1991], who is also unable to render useful grammars using a similar approach.

Threshold for number of sentences mutated

Section 7.3.1 detailed how we put a 50% chance of noisy channel mutation occurring on transferred structures. We have also conducted experiments in which this threshold was lowered to 25% and raised to 75% respectively, but to no avail. A GRAEL-2-BEST-type society in which the number of mutated structures was lowered to 25% (GRAEL-2(25%) in Table 7.15) performed worse compared to the 50% counterpart. It may however be the case that

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	Correct/*	%
Baseline	578	687	84.1	59.7	69.8	0/97	0.0
GRAEL-2-BEST	771	890	86.6	79.6	83.0	7/97	7.2
GRAEL-2(xover)	753	921	81.8	77.8	79.7	6/97	6.2
GRAEL-2(25%)	685	832	82.3	70.8	76.1	5/97	5.2
GRAEL-2(75%)	768	941	81.6	79.3	80.5	7/97	7.2

Table 7.15: Baseline and GRAEL-2 Accuracies on 97-sentence test set: Extra Experiments

the careful approach improves results should we allow the GRAEL-2 society to run longer. The society in which we raise this threshold to 75% (GRAEL-2(75%) in Table 7.15) performs worse as well, due to the large amount of noise that the over-eager mutation ratio seems to produce.

Sanity Check: Test Set Accuracy of All agents

An issue that has so far not been addressed is the halting point of the society. We spent a lot of time discussing the effects of the different halting procedures in GRAEL-1 and the so-called SELFITS to select the best candidate agent for parsing. But for the GRAEL-2 experiments we had to make do with looking at the understanding accuracies during language games to determine when to halt the society and which agent to select, because the larger test set required us to reserve all other structures for the training set.

It might be interesting to see if we would be able to squeeze out more performance from GRAEL-2 if we apply more intelligent halting procedures. The graph in Figure 7.10 might help us to provide an insight into this matter. This graph plots each agent’s accuracy on the test set throughout the entire lifespan of the GRAEL-20 BEST society. There are two horizontal lines in this graph: the bottom one displays the baseline accuracy of using the training set to parse the test set. The top line presents the F-score of the agent selected from the GRAEL-2-BEST-society at the understanding accuracy halting point to parse the test set. It is clearly shown here that the accuracy on the test set achieved by GRAEL-2-BEST is not exceptional: at each point during the

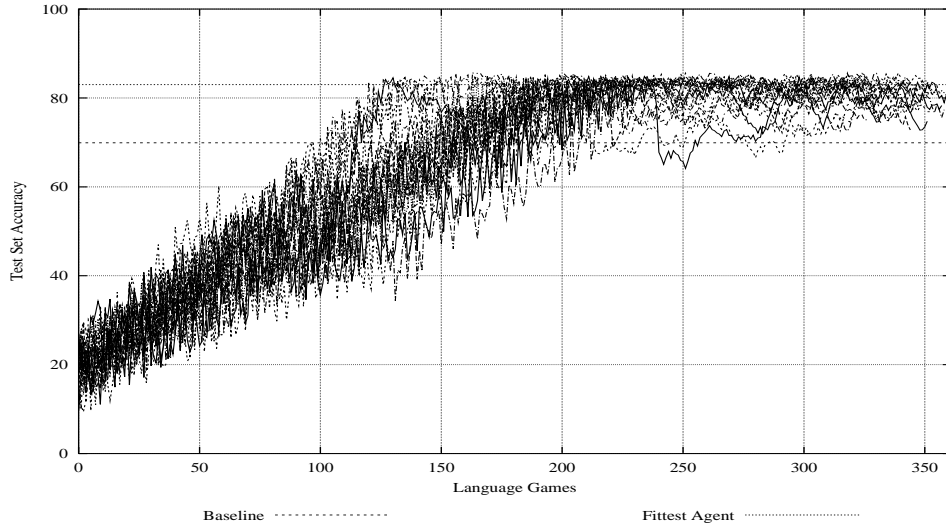


Figure 7.10: GRAEL-2-BEST - Test Set Accuracies

convergence phase of the society, there are a number of agents that achieve this kind of F-score. While there are many agents that achieve a considerably lower F-score (many of which “young agents” catching up to the rest of the society), there are also a few agents that overtake the top horizontal line, but not by a great margin.

This data shows that the score achieved by GRAEL-2 best is not some lucky shot achieved by a favorable random halting procedure. The best agents in the GRAEL-2-society achieve a good F-score for a considerable period of time, so that selecting the best agent is not as troublesome as in GRAEL-1. Judging from the graph in Figure 7.10, little is to be gained from implementing different halting procedures and SELFITS to boost performance. This does however indicate again that the difference between GRAEL-1 and GRAEL-2 amounts to more than just a variation on the same theme.

GRAEL-2 + GRAEL-1

This raises the issue of the difference between GRAEL-2 and GRAEL-1. The latter optimizes the probabilistic weights of grammatical structures in a grammar, but does not discover any new grammatical structures. The former

is able to employ the dynamics of the GRAEL-system to invent and evaluate new grammatical structures, but the data in Table 7.14 indicates that it is not at all able to optimize the probabilistic weights of the grammar, so that its overall parsing accuracies are similar to that of the baseline parser.

In Chapter 5 we explained the GRAEL-1-method as follows: by providing a society of agents with a “deficient” grammar and allowing them to improve on it by “practicing” their own grammars on other agents, the grammars become optimized in a setting which resembles the actual task at hand: parsing sentences. This view of GRAEL-1 provides an interesting window of opportunity to improve the results of GRAEL-2: GRAEL-2 leaves us with a collection of broad coverage grammars that is lacking in the distribution of probability mass. But GRAEL-1 provides a system that is geared towards such optimization in particular. Combining GRAEL-2 with a GRAEL-1 post-processing phase might therefore provide an interesting increase in results.

Figure 7.11 explains how the combination works. A training set of tree-structures is distributed over a GRAEL-2 society. The agents have a E-language consisting of a number of tree-structures and a I-language, containing a grammar induced from these trees. As GRAEL-2 progresses, the E-language trees are changed, while the I-language obtains a lot of new grammatical information.

Using the understanding halting procedure, we then halt the society and distribute the original tree-structures over the agents again, who add them to the mutated structures in their respective E-languages. This provides the agents with a gold-standard touchstone of tree-structures from the original training set, while the mutated structures are maintained as a criterion for the mutated grammar rules in the I-languages. After that, the society progresses as a GRAEL-1 society and no more structures are mutated and no interaction between I-language and E-language is allowed. The society is halted when the understanding accuracies are leveling out again.

We conducted two experiments: one is trained and tested on the standard GRAEL-2 division, while the other experiment is conducted on the GRAEL-1 division and was tested on a 58 sentence test set. Figure 7.12 displays the course of the experiment on the GRAEL-2 division (the experiment with the GRAEL-1 division ran a similar course). This graph quite noticeably illustrates the point at which GRAEL-2 turns into GRAEL-1. Overall under-

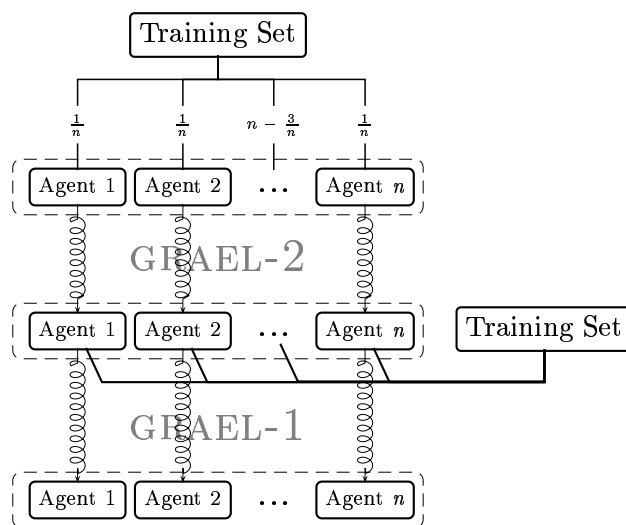


Figure 7.11: Combining GRAEL-2 and GRAEL-1

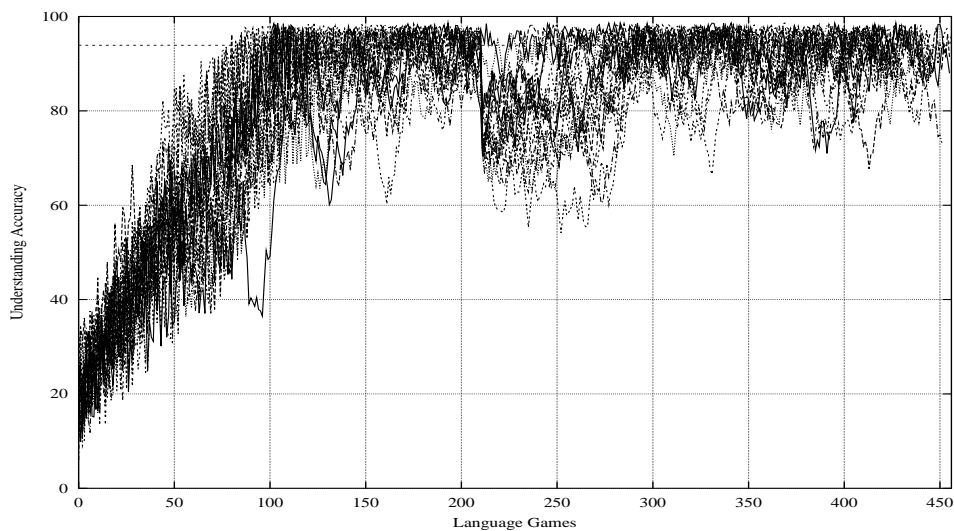


Figure 7.12: GRAEL2+1 Experiment - Understanding Accuracies

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/968)	%	/97	%
Baseline	578	687	84.1	59.7	69.8	0	0
grael-1	624	712	86.3	64.5	73.8	0	0
grael-2	771	890	86.6	79.6	83.0	7	7.2
grael1+2	801	902	88.8	82.7	85.7	11	11.3

Table 7.16: GRAEL2+1 Experiment - GRAEL-2 Division - Results

standing accuracies drop a little, but as GRAEL-1 progresses, understanding accuracies rise again.

Table 7.16 displays the results of the combination of GRAEL-1 and GRAEL-2 on the 97-sentence test set used in the main GRAEL-2 experiments. The results are very encouraging: the combined system achieves an F-score of **82.7%**. The increase shows that GRAEL-1 has indeed redistributed the probability mass so that the grammar is better tuned to parsing sentences, rather than exploit GRAEL-2's predilection for coverage. The most impressive gain is seen on exact match accuracy: 11.3%. Even though this is still a modest result, the increase over the standard GRAEL-2 approach is clear.

Table 7.17 shows results on the standard GRAEL-1 test set. The GRAEL-2 approach achieves an F-score that is similar to that of the baseline model. This is an almost identical situation as we had observed during the main GRAEL-2 experiments. GRAEL-1 outperforms GRAEL-2 by a significant margin, while the combination of GRAEL-1 and GRAEL-2 further increases the F-score over GRAEL-1. Note however that in absolute terms, there is only a difference of seven constituents between the two systems, although this figure belies more fundamental differences. These differences are projected in the exact match accuracy, which increases to 86.2%. The mutated information that GRAEL-2 brings to the mixture made it possible for 12 previously unparsable sentences to be parsed, of which five correctly.

This experiment that combined GRAEL-1 and GRAEL-2 proves that the two systems are very different from another but this difference can be used to each system's advantage. The results show that GRAEL-2, although modest in its results, can provide a grammar that expands the coverage, while GRAEL-1 takes care of the fine-tuning of probabilities. The combination of

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/496)	%	/58	%
Baseline	437	483	90.5	88.1	89.3	41	70.7
grael-1	454	491	92.5	91.5	92.0	45	77.6
grael-2	443	495	89.5	89.3	89.4	43	74.4
grael1+2	461	498	92.6	92.9	92.8	50	86.2

Table 7.17: GRAEL2+1 Experiment - GRAEL-1 Division - Results

the two therefore provides a way to extend an original corpus-induced grammar to a large coverage grammar that is optimized for the task of parsing unseen data.

7.4 Experiments: Wall-Street-Journal

Due to the complex nature of processing a GRAEL-2 society⁸, experiments on the WSJ-corpus are limited to one experiment using the standard division: Sections 1 to 21 as a training set and Section 23 as a test set. The same adjustments were made on the GRAEL-system as in the GRAEL-1 WSJ-experiment:

- No validation set was used. The halting point and fitness of agents was determined by looking at inter-agent communication
- 40 slices of 1.000 sentences were brought into the system at a 40 run interval

We conduct a single GRAEL2+1 experiment using a 100-agent crossover-based society. Interaction between I-language and E-language is included⁹ and mutation only occurs as noisy channel mutation with the weighted mutation combination **Adc**. Figure 7.13 shows the course of this experiment.

⁸The interaction between I-language and E-language entails a doubling of the amount of parsing done per language game run.

⁹To speed up parsing, a 50% sampling of the I-language was used to construct the E-language.

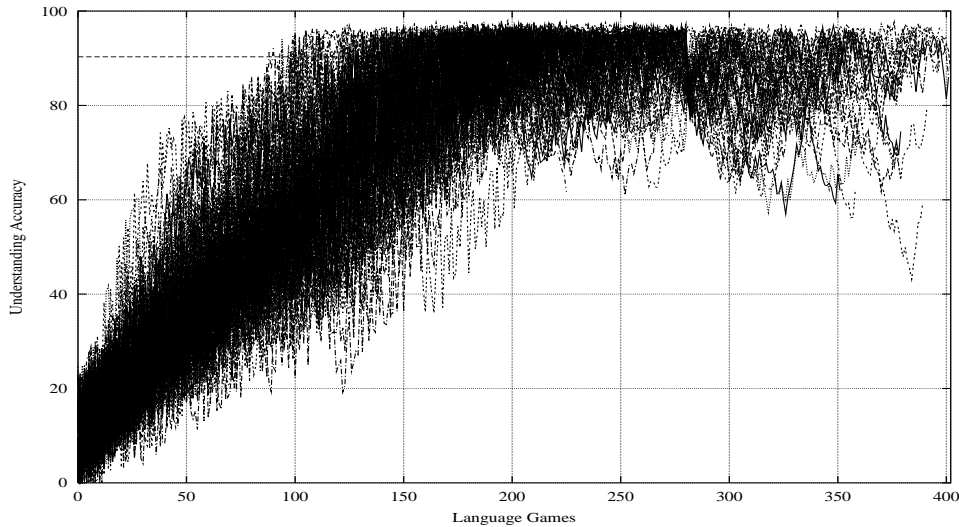


Figure 7.13: GRAEL2+1 Experiment - WSJ - Understanding Accuracies

The transition between GRAEL-2 and GRAEL-1 is slightly visible at language game run 281. The plots for GRAEL-1 continue at first in a more dispersed state, but a 2nd halting point (i.e. a plateau in understanding accuracy) is reached, when some agents' slots have used their 10 generations and the number of agents decreases.

The results of this experiment can be read off Table 7.18. GRAEL-2 (right before the transition to GRAEL-1) does not perform well on the WSJ-corpus and is well below baseline. Yet, it does prove to be able to provide good mutated rules: [Collins 1999] states that there are 17.1% of sentences in the test data that have a rule not seen in the training data. In our version of the WSJ-dataset, this percentage is 20.1%. The fittest agent's grammar in the GRAEL-2-society reduces this ratio to 18.3%, while the grammar induced from the entire GRAEL-2-society reduces it further down to 17.4%. Combined with GRAEL-1 this causes a .2% increase on the F-score, while 14 more sentences are parsed correctly.

	Correct	LP		LR	$F_{\beta=1}$	ExMa	
		/	%	% (/496)	%	/58	%
Baseline	37535	45908	81.8	79.3	80.5	386	16.0
grael-1	38401	46985	81.7	81.1	81.4	551	22.8
grael-2	36355	47658	76.3	76.8	76.5	467	19.3
grael-2+1	38549	47102	81.8	81.4	81.6	565	23.4

Table 7.18: GRAEL-1, GRAEL-2 and GRAEL-2+1 experiments on WSJ-corpus

7.5 Advances and concluding remarks

The experiment on the WSJ-corpus provides results that encourage many extensions to the research described in this chapter. There are still numerous experimental parameters left to optimize, but the same conclusion we drew with respect to the GRAEL-1 experiment holds here as well: currently the computational resources are lacking to fully test all experimental parameters of GRAEL-2. Extensions of the research should include some dynamic way to determine the best weights for the combination of mutation operations, which was determined rather ad hoc in this chapter.

In the context of the experiments that varied the threshold of the number of structures that were mutated, we also suggested that a lower threshold of 25% might provide an increase if we leave the GRAEL-2-society running for a long period of time. Future research should therefore look into an automatic method for determining the best balance between the ratio of mutated structures and the life-span of a GRAEL-2-society.

The beneficial combination of GRAEL-2 and GRAEL-1 also provides a window to future research investigating a less abrupt transition from GRAEL-2 and GRAEL-1. Perhaps some effort should be invested in finding a type of GRAEL-society that integrates the 2. The interaction between I-language and E-language seems to be a crucial issue in this matter: its absence is required for grammar optimization, but only its presence can produce new rules. A way to combine the two could be to allow the interaction in some agents and prohibit it in others. The ratio of each agent type however, would need to be experimentally established.

In this chapter we presented an instantiation of the GRAEL-environment that was able to create new grammatical structures and assess their usefulness in an agent-based evolutionary environment. Evaluating a grammar rule induction method is in essence an empirical problem: its purpose is to create grammars that provide a broad coverage for grammatical structures, so that marginal and unusual structures, that were previously unavailable can now be parsed by the grammar. But this is exactly the problem: previously unavailable structures are just that: unavailable. This makes it hard to manually inspect and evaluate a mutated treebank grammar, as one can never be sure what rules could need to be triggered in what context.

The test set we compiled to perform the GRAEL-2 ATIS-experiments on went a long way in providing a decent touchstone to see how well GRAEL-2 performed as a supervised grammar induction/rule discovery method. And results indicate it seems to perform quite well: mutated information becomes available that is able to create parses for difficult constructions, while the number of structures that constitute noise is limited and is attributed a small enough portion of the probability mass as not to stand in the way of actual useful mutated structures.

The key feature towards the success of GRAEL-2 is the addition of interaction between I-language and E-language, which we ruled out as a beneficial setting for GRAEL-1, which required the actual correct tree-structures to render an optimal distribution of probability mass. Since the E-language serves as a benchmark for other agents, it provides a window of opportunity to test the validity of mutated structures. The interaction between I-language and E-language then provides feedback on the quality of mutated structures in two ways. First, agents will have a harder time incorporating nonsensical structures in the analyses for the sentences of their E-language, so that there will be a preference for cleverly altered structures, if not for the original grammatical structures. And second, the mutated structures that are eventually featured in the E-languages are spread throughout the society, to be picked up by another agent who will process it in his I-language and possibly use it for future reference, if its usefulness can be confirmed.

The addition of this type of interaction does counteract GRAEL-1's ability to optimize the distribution of the probability mass of a grammar however and a grammar obtained from a GRAEL-2 society therefore seems unsuited to be directly applied to parsing. But combining it with a GRAEL-1 society

however, goes a long way in resolving this issue, providing a grammar that has a broader coverage, as well as a better tuned probability mass distribution over the structures contained therein.

11:15, restate my assumptions: 1. Mathematics is the language of nature. 2. Everything around us can be represented and understood through numbers. 3. If you graph these numbers, patterns emerge. Therefore: There are patterns everywhere in nature.

Lenny Meyer - Pi - ©1998

8

GRAEL-3: An Agent-Based Evolutionary Computing Approach to Unsupervised Grammar Induction

We described a parser using a probabilistic grammar induced from a training set of annotated tree-structures in Chapter 3. The constituent structure, i.e. the segmentation and labeling properties, were directly derived from examples in the training set, and the probability mass in the grammar mirrored that of the original training set. In Chapter 5 we used an agent-based evolutionary computing method to redistribute the probability mass so that it reflects probabilities needed for the actual task of parsing. GRAEL-1 did however not alter the content of the structures itself.

Strictly adhering to the segmentation and labeling properties that the treebank provides us, gives rise to problems of *grammar sparseness*, when grammatical information needed to parse the test set is not featured in the training set. Chapter 7 described a method to mutate existing grammatical structures and evaluate them in a GRAEL-2 society. This constituted a

workable grammar discovery rule method. From a slightly different point of view, GRAEL-2 could also be conceived as a supervised grammar induction method, since new grammatical information is created on the basis of actual structures in an annotated corpus. This provides a guideline for segmentation and labeling to which the newly created structures to a large extent still adhere to.

In this chapter we take one step further and abandon all pre-defined grammatical information in the GRAEL-environment. Grammatical structures are provided to the agents in a GRAEL-3 society on the basis of information theoretic distributional properties between words (or tags) in previously observed sentences. By consequently playing a large number of language games, the agents in the society adapt to each other's grammars and optimize them for parsing. This effectively establishes GRAEL-3 as an unsupervised grammar induction method, as it induces and optimizes a grammar on the basis of raw, unstructured textual data.

We will take a look at some relevant research efforts in the field of unsupervised grammar induction in Section 8.1. Adapting some of the insights of this research to the GRAEL architecture allows us to develop GRAEL-3 in Section 8.2, after which we describe the experimental setup and results of the experiments in Section 8.3. We conclude by discussing the capabilities of GRAEL-3 as an unsupervised method for grammar induction in Section 8.4.

8.1 Unsupervised Grammar Induction

In Chapter 7 we discussed some genetic programming approaches for grammar induction [Dupont 1994; Huijsen 1993; Kammeyer and Belew 1996; Keller and Lutz 1997a; Keller and Lutz 1997b; Lankhorst 1994; Lucas 1993; Lucas 1994; Wyard 1989; Zhou and Grefenstette 1986; Losee 1995; Blasband 1998] and stated that most of these systems employ evolutionary computing to induce a set of grammar rules that can cover a small, artificially constructed set of sentences. The data sets most of these systems used, were such that the grammars could not be extrapolated to realistically sized corpora.

But outside the field of genetic programming, a large body of work has

researched the unsupervised induction of grammars using different kinds of algorithms, like neural networks [Honkela et al. 1995], Bayesian methods [Stolcke and Omohundro 1994; Chen 1995], but mainly using concepts from information theory [Magerman and Marcus 1990; Grunwald 1994; de Marcken 1995; Yuret 1998; M. Redington and Finch 1998; Wolff 1998; Clark 2001]¹. [van Zaanen 2002] describes the interesting method of using alignment based learning to bootstrap the acquisition of structure. Like [Adriaans 1999], it is based on the notion of substitutability described in [Harris 1951].

A discussion of all methods for unsupervised grammar induction would fall beyond the scope of this chapter, so we will therefore limit ourselves to a brief discussion of some systems based on information theory, that are relevant to the grammar-bootstrapping method employed by the GRAEL-3 system. We will also refer to [van Zaanen 2002] and [Clark 2001] in Section 8.3.1, as they comprise the first methods evaluated on a purely objective basis. But first we will discuss Lexical Attraction modeling as conceived by [Yuret 1998].

8.1.1 Lexical Attraction Modeling

[Yuret 1998] starts out by stating that his approach to unsupervised grammar induction does not find grammatical relations in the way a phrase-structure grammar formalizes them. He argues that a ps-grammar *only indirectly represents [linguistic] relations as side-effects of the constituent-grouping process* and therefore turns to dependency grammars [Mel'čuk 1988; Sleator and Temperley 1991] as his formalism of choice.

Figure 8.1 compares a PS-type structure to a dependency representation. A phrase-structure can be interpreted in two ways: in a bottom-up interpretation, a phrase structure tries to group terminals into constituents, after which the superordinate structures are grouped into constituents of their own, until finally one top-level node is reached. In the top-down point-of-view, a structure starts with a single node, to be expanded in such a way as to cover all terminals. A dependency structure on the other hand always needs to be considered in a bottom-up fashion. It links words related to one

¹Thanks to [van Zaanen and Adriaans 2001] for a good overview of the most relevant research efforts in the field of unsupervised grammar induction.

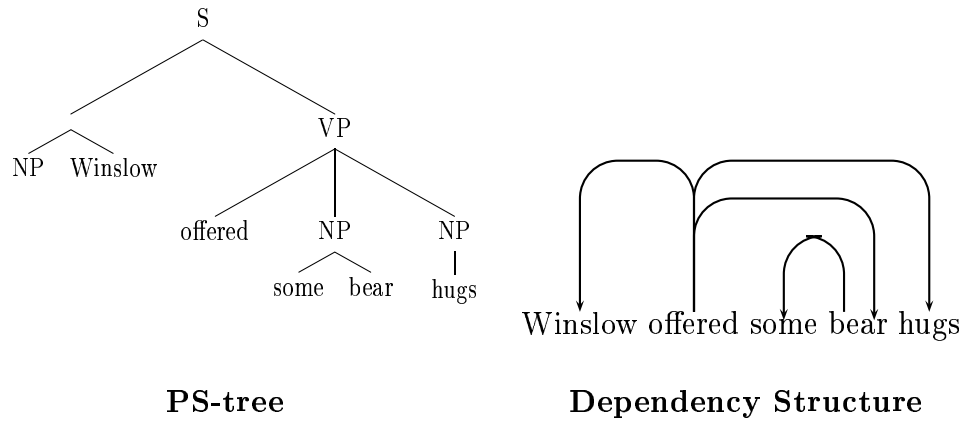


Figure 8.1: PS-tree vs Dependency Representation

another directly and formalizes their dependency relation.

[Yuret 1998] argues for the case of using a dependency formalism for his unsupervised grammar induction task as follows:

First, the indirect representation of phrase-structure makes unsupervised language acquisition very difficult. Second, if the eventual goal is to extract meaning, then syntactic relations are what we need, and phrase-structure only indirectly helps us retrieve them [Yuret 1998] (p.21)

[Yuret 1998] also states that there are many other accounts that state that the automatic induction of phrase-structure grammars is very problematic [de Marcken 1995], if not impossible [Gold 1967]. [Yuret 1998] therefore adopts the dependency formalism to represent the structure to be induced and claims that dependency between words can be measured by looking at the *affinity* between words, or in other words: their **lexical attraction** to one another. The lexical attraction method which [Yuret 1998] proposes, tries to express the dependency between words by using concepts from information theory.

Consider the following sentence (example reproduced from [Yuret 1998]):

The IRA is fighting British rule in Northern Ireland

We can express the probability of the words in this sentence, by referring to a large corpus of sentences and looking at the likeliness of each word occurring according to a unigram-model, which looks at each word in isolation, using the following formula:is

$$p_w = \frac{n(w)}{n(*)}$$

w = focus word

* = any word

n(*i*) = function counting the number of times *i* occurs

Following [Shannon and Weaver 1949] we can compute the information content of the context word, by looking at the amount of bits needed to encode the word. This entropy figure can be calculated on the basis of its probability (cf. supra) using the formula: $-\log_2 p_w$. We can now attribute each word in the example sentence an entropy figure expressing its information content according to a unigram model:

The	IRA	is	fighting	British	rule	in	Northern	Ireland
4.20	15.85	7.33	13.27	12.38	13.20	5.80	12.60	14.65

The information content of the sentence can be computed, simply by adding the information content figures of each individual word: 99.28bits, which equates to a $2^{-99.28}$ probability of encountering this sentence, based on the probabilities induced from the corpus.

Yet, unigram models do not take any kind of context into account and are therefore considered as weak language models that do not capture any kind of relevant linguistic information. If we expand the scope of our model and consider a bigram model, which calculates the probability of a word, based on the preceding (or possibly the following) word, we can induce some useful information: in our example, the word *Northern* carries 12.6 bits of information by itself. Yet, in our data set, it precedes the word *Ireland* 36% of the time. Consequently, the word *Northern* only adds 1.48 bits of new information to *Ireland*. Using this information, we can recalculate the number of bits needed to encode the phrase *Northern Ireland*: using a bigram model this phrase carries 16.13 bits (14.65 + 1.48) as opposed to 27.25 bits (12.60 + 14.65) in the unigram model.

We can now re-assign information content measures to the words in our sentence, using a bigram model:

The	IRA	is	fighting	British	rule	in	Northern	Ireland
4.20	12.9	3.73	10.54	8.66	5.96	3.57	9.25	3.53

We now only need 62.34 as opposed to 99.28 bits to encode this sentence. This exemplifies how a bigram-model will almost always require less bits to encode the same sentence than a unigram model. Following the maximum likelihood principle, which tries to maximize the probability of a sentence (and consequently minimize its entropy), the bi-gram model can be considered as the better language model.

[Yuret 1998] now argues that we can further reduce the entropy by interpreting the context of a word in terms of its syntactic relation to other words rather than just looking at the surrounding words. The information content of a word is then calculated on the basis of its head or modifier in a dependence structure. Lexical attraction tries to retrieve these syntactic relations, by linking words to each other in such a way that the amount of bits required to encode a sentence is minimized. In other words: all combinations of words are considered and only that combination of links, i.e. dependencies, is maintained which minimizes the amount of bits needed to encode the sentence.

Determining which links are maintained and which links are discarded, depends on their mutual information content, as well as on the formal restrictions the dependency formalism imposes on them: links between words have to be acyclic² and planar³. [Yuret 1998] hypothesizes that, given enough data, the dependency structure this linkage provides is the correct one for the sentence, since the affinity of words in a sentence can be directly measured in terms of their mutual information content.

To compute the mutual information between two words, [Yuret 1998] considers combinations of words and computes their mutual information with the following formula:

²Each word is linked to only one head, except for the head word that governs the entire sentence.

³Links cannot cross.

$$MI(x, y) = \log_2 \frac{P(x,y)}{P(x,*)P(*,y)} = \log_2 \frac{n(x,y)/N}{(n(x,*)/N)(n(*,y)/N)} = \log_2 \frac{n(x,y)N}{n(x,*)n(*,y)}$$

MI: mutual information

$P(x,y)$: probability of considering x y as a combination

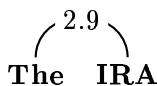
$n(x,y)$: the count of (x,y)

N : total number of observations made

There are three different methods for considering combinations: the first method only records (1) adjacent pairs, the second method records (2) all possible pairs and the third method uses (3) combinations of pairs identified by the processor. The latter method requires some explanation: if a sequence of words AXB is found, the first method records the adjacent pairs A-X and X-B. If the processor discovers a sentence like AX...YB, it will now consider the combinations A-Y and X-B. Because of the planarity restriction, no linking between words X and B is allowed, unless they can break the A-Y and X-B links. After these combinations are recorded, more structures can be built using the recorded information and so on so forth.

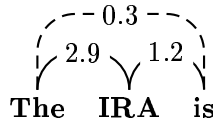
In practice however, only a limited number of possible links are considered and not all possible linkages are computed. [Yuret 1998] describes an approximation algorithm, which processes sentences from left-to-right, considering links for the current word with each of the preceding words. This approximation does not guarantee to find the most likely linkage, but experimental results show that little is to be gained from employing the optimal algorithm, that considers both directions.

Returning to our example enables us to describe how the approximation algorithm works. The first two words of the sentence our processed and their mutual information is computed:

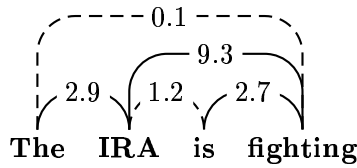


The IRA

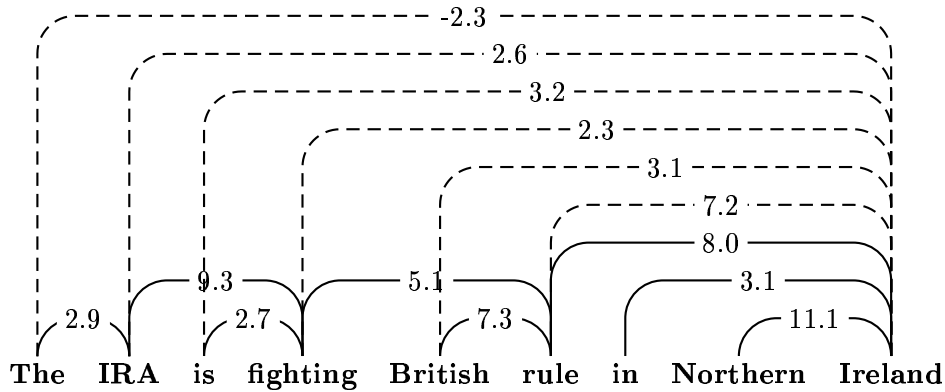
The following word enters the processor. Two possible links are considered now:



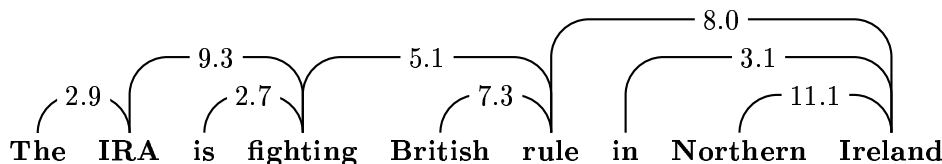
This introduces a cycle however: *the* has two heads (cf. dashed line), which violates the formal aspects of dependency grammars. Therefore the weakest link in the cycle is removed and the next word enters the processor:



Cycles are removed and the strongest links are maintained in this structure. This processing goes on for all words in the sentence until finally we reach the last word of the sentence, which brings us to the following situation:



Again a lot of links are considered, the introduction of which would cause cyclic dependencies (cf. *Ireland-rule*) or crossed links (cf. *Ireland-is*). The weaker links are removed, which finally leaves us with the following dependency structure:



This is the correct dependency structure for the sentence, as found by the lexical attraction model. The actual lexical attraction model was trained on a 100 million words data set and tested on a held-out set of 200 sentences, in which the content words were linked (1287 links). The test set featured a restricted vocabulary, since linking of unknown words is not supported by the lexical attraction model. Lower bound precision/recall scores⁴ are 8.9/5.4%, while there is an upper bound of 85.7% recall as this is the ratio of links that actually featured positive lexical attraction between them.

Using different methods for recording pairs (cf. *supra*), the lexical attraction models are able to achieve a precision score between 75%(method (1)) and 55%(method (2)) and a recall score ranging between 40%(method(1)) and 50%(method (3)). Unfortunately [Yuret 1998] only included experimental results plots, rather than numbers, so that these figures are only an estimate.

These results are modest, but encouraging. The lexical attraction method is indeed able to retrieve affinity between words and meaningful dependencies from this information. The advance of lexical attraction is however limited by the fact that it does not provide a label for the syntactic relations that are retrieved and does not generalize over the data in the way a ps-grammar induction method is able to.

[Yuret 1998] states that dependency formalisms are able to capture dependencies between words, such as subject-verb relationships, that can only be indirectly induced from a phrase structure. But what is lacking from the output of the lexical attraction model, is exactly the nature of the syntactic relations that are retrieved, so that the most detailed representation this approach can yield is the description of what words are syntactically related (cf. the dependency structure in Figure 8.1). In no way is there however any

⁴These were obtained by providing random numbers when the lexical attraction of a combination was measured.

indication of what kind of relation has been established. Since the nature of a syntactic relation in the dependency structure would need to be indirectly retrieved from this type of structure as well, this renders Yuret's argument against ps-structures as a representation method moot. Even though dependency structures do have distinct advantages over phrase structures, [Yuret 1998] does not exploit them.

Furthermore, it is also clear from Figure 8.1 that a phrase structure may contain semantically relevant information expressed in its higher-order structures that a dependency formalism can only capture indirectly. In the right hand structure in Figure 8.1, it can be deduced that *Winslow* is the one that is *offering* something, whereas the phrase structure displays clearly what it is that *Winslow* is *offering to whom*.

Since each formalism seems to have its advantages, which renders things more or less equal from a theoretical point of view, the grammar bootstrapping method for GRAEL-3 will try to induce phrase structures. The method that is used, however, is based on the same principles employed in [Yuret 1998] to create basic dependency structures. Despite its fundamental shortcomings, lexical attraction does provide an interesting minimalist approach to building structures in an unsupervised manner.

8.1.2 Other Distributional Methods

Yuret's reluctance to deal with the unsupervised induction of ps-grammars is the result of a long tradition of troublesome unsupervised grammar induction techniques. The underwhelming results achieved by applying an inside-outside algorithm for grammar induction [Baker 1979; Lari and Young 1990] have prompted researchers to simplify the task by using a partially labeled data set [Pereira and Shabes 1992; Briscoe and Waegner 1992]. As [de Marcken 1995] rightfully points out, this benefits the engineering task of unsupervised grammar induction, but it does not establish an actual method for the unsupervised induction of grammar, as it involves human annotation effort on raw data.

[de Marcken 1995] explains the limited success of information theory-based approach to ps-grammar induction by pointing out that a phrase structure does not necessarily establish a low-entropy representation of a sentence.

Using information theoretic measures on sequences of words to compile a ps-grammar is therefore almost guaranteed to fail.

Unsupervised methods for the induction of ps-grammars are very vulnerable to local maxima, as early decisions on low-level structures made by the induction algorithm are unlikely to be reversed during later processing stages. [de Marcken 1995] suggests that either a different search strategy is needed to induce ps-grammars, or an entirely different representation scheme to represent syntactic structure altogether.

Whereas [Yuret 1998] adopted the latter solution, [Clark 2001], a recent attempt to apply information theory concepts on real unsupervised induction of ps-grammars, redefined the notion of mutual information and therefore seems to have implemented the former solution. [Clark 2001] uses a window of three tags: the focus word, the preceding and the following word and records these sequences and their respective information content. Constituent boundaries can then be determined by looking at the mutual information of these sequences: [Clark 2001] hypothesizes that if a high mutual information value exists between the symbols immediately before and after a constituent candidate, the two symbols are not independent.

To counter the overestimation of mutual information for sparse data, [Clark 2001] normalizes the mutual information measure on the basis of the distance between them. A grammar can be induced by applying a minimum description length approach: an initial grammar is created that assigns one rule for each sentence type. Next frequent collocations are clustered and the mutual information criterion filters out the spurious constituents. Next, a cluster is selected that provides the largest reduction in description length and a new non-terminal is added with rules for each sequence in the cluster. Finally, the new rules are created to perform a partial parse of the sentences after which new sequences are looked up.

[Clark 2001] tested the system on the ATIS-corpus, an evaluation method for unsupervised grammar induction proposed by [van Zaanen and Adriaans 2001], which we will discuss in some more detail in Section 8.3.1. The results are very encouraging with a **42.0%** F-score on the ATIS-corpus. [Clark 2001] proves that a system based on information theory for the induction of ps-grammars is not a *contradictio in terminis*. [de Marcken 1995] pointed out that there is an inherent danger attached to these systems of getting stuck in

a local maximum, but [Clark 2001] seems to resolve this by defining mutual information in terms of the symbols *surrounding* the putative constituent. Using this information to filter out spurious constructs, minimum description length measures seem less vulnerable to the aforementioned problem that the induction of ps-grammars tends to get stuck in a local maximum.

8.2 GRAEL-3

In this section we will discuss the development of an unsupervised grammar induction method that can be used to bootstrap and process a GRAEL-3 society. We will first commit to a representation model for compositionality in language (ps-grammar) and a method to infer this from raw data (lexical attraction), after which we will describe how grammars can be induced and optimized using our distributed evolutionary computing approach.

8.2.1 PS-grammar as a performance model

A lot of time and effort has been spent by formal linguists, psycholinguists and computational linguists alike arguing against the case of ps-grammars as a suitable formalism to describe structure in language. While there are many valid arguments for discarding ps-grammars, none of the alternatives succeeds however in providing a formalism with as much impact on the field of linguistics as the formalism proposed in [Chomsky 1957]. Present day parsing algorithms are still largely based on the early algorithms that try to incorporate ps-grammars in computational implementations. And while a considerable amount of computational linguists try to implement other formalisms with varying degrees of success, the implementations hardly ever seem to be expanded in optimized engineering approaches towards parsing, especially compared to parsers for ps-grammars. Furthermore, researchers trying to compile a corpus of annotated structures (cf. [Marcus et al. 1993]), are more likely to prefer a controversial, but well established grammar formalism in favor of a more obscure, but psycho-linguistically more relevant theory.

Most research in the field of machine learning of natural language syntax

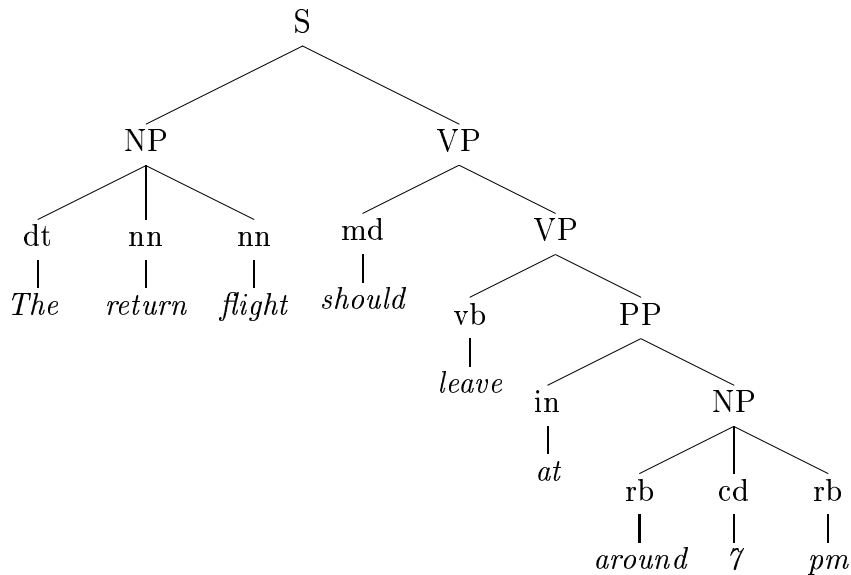


Figure 8.2: A phrase structure for *The return flight should leave at around 7 pm*

has conformed with ps-grammars to some extent. Even a task like shallow-parsing, which typically does not try to build full ps-trees for a sentence, is firmly rooted in the segmentation and labeling aspects that the ps-formalism provides. Phrase structures do indeed seem able to provide the kind of segmentation for basic constituents that seems acceptable for most researchers and even the more controversial higher-order grouping of constituents can provide tree-structures from which a decent amount of semantic information can be deduced.

Consider the example in Figure 8.2. This structure identifies a topic *the return flight* and a focus *should leave at around 7 pm* and expresses this structurally. Semantically valid clusters of words are grouped in the same constituent structurally (*at around 7 pm*), while the scope of the modal modifier *should* is correctly displayed in the structure it governs. The example shows that, although this structure does not explicitly display subject/object distinctions or subcategorization restrictions, the most basic structural properties of the sentence can be described in a phrase structure.

Its greatest strength from an engineering point of view is that it presup-

poses a minimal set of assumptions on the words: trees can be constructed, almost regardless of the actual content of the words, simply on the basis of their distributional properties. Parsing in this view does not require language understanding proper and syntax is viewed as an expressive module in its own right.

We would like to argue that this may be its greatest strength from a linguistic point of view as well. If we consider syntactic structure as a necessary precursor to the extraction of meaning from a string of words, explicitly describing syntactic structures in terms of semantic relationships between words (cf. dependency grammar, HPSG,...) reduces the functionality of syntax as an entity expressing meaning by itself, as it has been reduced to a mere by-product of semantic relations between the words of the sentence. In other words, it does not carry any meaning by itself, it simply provides a structural representation for it. But even though postulating this direct link between semantics and syntax is highly relevant towards explaining language understanding, many of the formalisms incorporating this relationship are not. Let us for instance turn back to the dependency grammar formalism: it indicates (semantic) relations between words in a sentence which is fine if we want to explain how structure arises from meaning. But if we want to hypothesize that structure attributes meaning (e.g. through word order), a ps-grammar is indeed a formalism that presupposes a minimal set of assumptions and is therefore suited to represent syntax proper.

Our decision to maintain phrase structures as our syntactic representation of choice is motivated by this (undoubtedly controversial) assessment of ps-grammars from a linguistic point of view. If the reader disagrees with this motivation, (s)he can disregard the linguistic motivation and focus on the engineering point of view, which simply requires us to use ps-grammars *as the Romans do*, to enable a comparison with other grammar induction methods. This avoids the problematic evaluation of the grammar induction system that was apparent in [Yuret 1998].

Given the historical development of ps-grammars into Government and Binding type theories, we need to explain what exactly our definition of a ps-grammar is, as most of those theories obviously do not conform to our view of ps-grammars as a formalism making a minimal set of assumptions. The following issues are paramount to our view of ps-grammars as a minimally presupposing grammar formalism:

- A ps-grammar uses a set of ps-rules for segmentation and labeling: a string of words is segmented into constituents, by matching the right-hand side of a re-write rule in a grammar. The category on the left-hand side of the rule is used as the label for that segment. The category label should provide a minimal description of the items in the constituent.
- A ps-grammar provides a structure for a sentence, expressing **structural** properties first and semantic properties as a by-product of the structural properties.
- A phrase structure grammar should **not** require subcategorization information of the words in the sentence to yield a structure.
- A ps-grammar in our view constitutes a **performance model** of language, not a competence model. It should be able to provide syntactic structure for any kind of utterance made by language users.
- The use of phrase structures to represent syntactic properties **does not** presuppose nativism (see Chapter 10).

These key issues greatly reduce all prior assumptions that are usually connected to using a ps-grammar. In our view, it is a suitable approach to minimally describe full structures for sentences, using a simple segmentation and labeling approach.

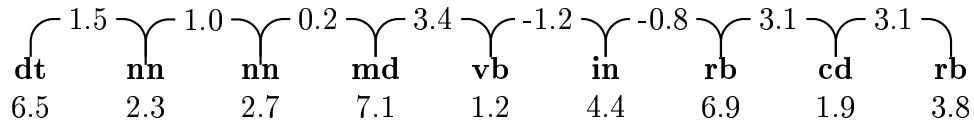
8.2.2 Information Theory for grammar induction

In Section 8.1.1 we introduced some basic concepts from information theory and showed how [Yuret 1998] is able to exploit this information to induce dependency structures from raw text. We discussed some of the difficulties researchers encounter when using this approach to induce a ps-grammar in Section 8.1.2. But [Clark 2001] showed that it is possible to use mutual information values to induce as well as filter grammar rules. In this section, we would like to describe a simplified version of the method described in [Yuret 1998] for the induction of ps-grammars that does not require massive data sets or several processing stages like [Yuret 1998] and [Clark 2001] do.

With GRAEL-3, we wish to develop a method to induce grammars for natural language. Whereas in GRAEL-1 and GRAEL-2 the agents were provided with a set of pre-parsed examples on which to base the development of their grammar, this kind of information is not available to the agents in GRAEL-3. We can supply the agents with natural language sentences, but they will need some grammar induction method to bootstrap their grammatical knowledge about them. The system described in this section will establish such a method.

The method we propose is based on information content values provided by a simple bigram-model expressing “lexical attraction” between words in a bigram. Given a set of sentences, the bigrams are recorded (in two passes: from left-to-right and from right-to-left), so that mutual information between adjacent words is recorded (cf. Section 8.1.1).

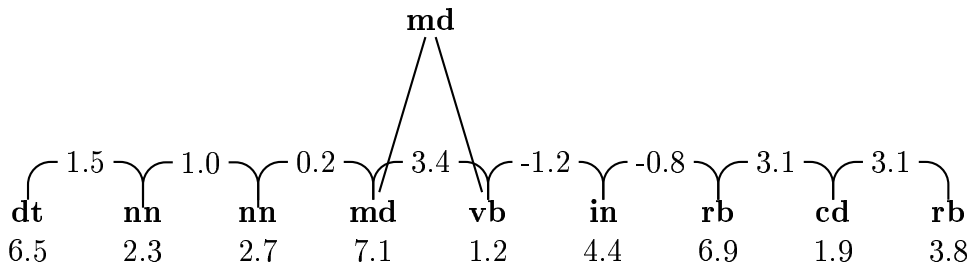
Let us consider an example for a grammar induction task that works for part-of-speech tag sequences of the ATIS-corpus. We are given a part-of-speech tag sequence for the sentence “*the return flight should leave at around 7 pm*”:



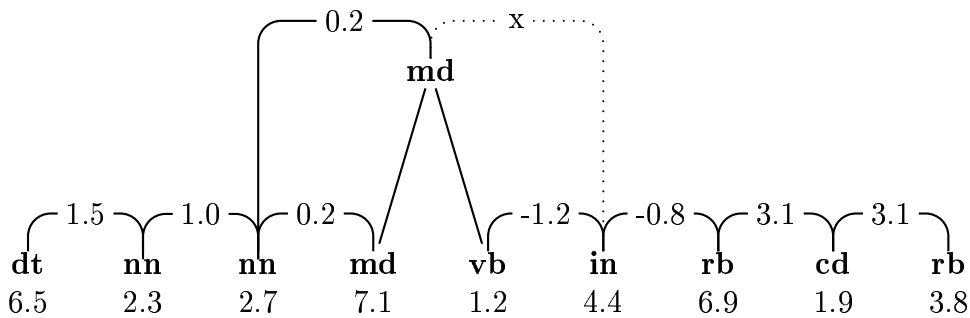
Each part-of-speech tag is attributed a value expressing its information content based on the preceding word⁵. The mutual information between two adjacent words is expressed on the arcs between part-of-speech tags. This provides a very rough idea of the affinity between the words on which we can build our syntactic structure.

Using this information, we now greedily lookup the bigram that has the highest mutual information content and join it together in one constituent bearing the label of the part-of-speech tag with the highest information content, to ensure that it is reflected in the labeling properties of higher order structures:

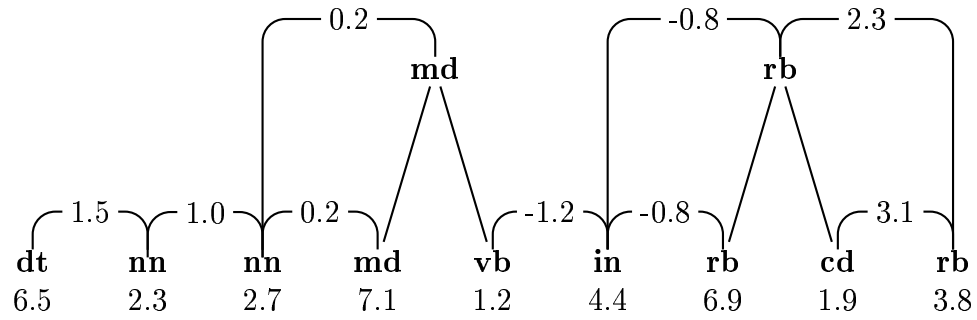
⁵The figures displayed in the example express actual values measured on the entire ATIS-dataset.



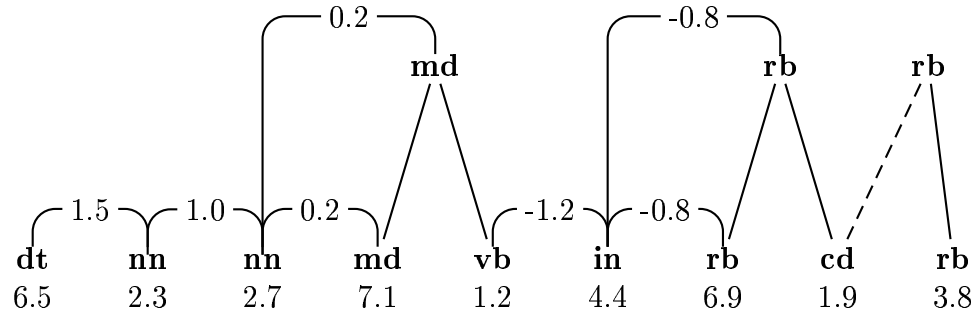
We now consider the newly created constituent as one symbol and link the preceding and following tag to this node as if it were in fact a single tag. The sequence “**md in**” has never been observed in the corpus and there is therefore no lexical attraction between these tags and the link will be discarded in the next step. Note that the 2nd **nn** can be linked to the **md**-terminal node directly or to the superordinate **md**-node:



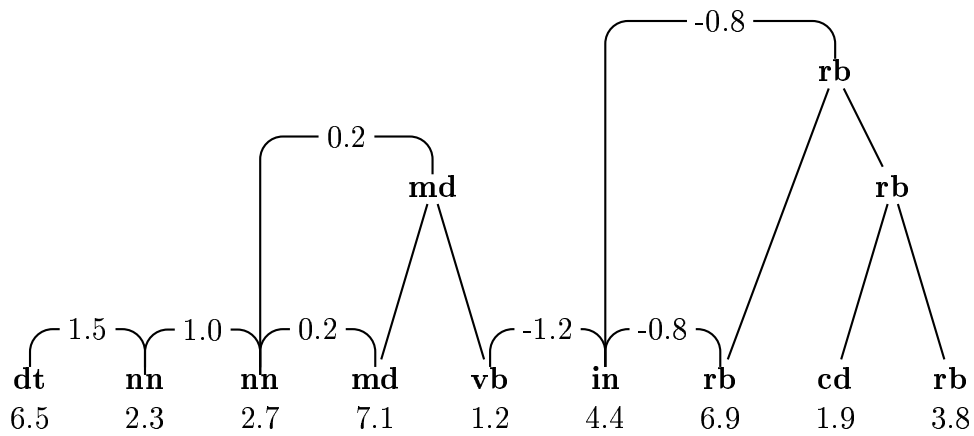
After this first grouping, we greedily look for the next case of strong lexical attraction and find it in two sequences “**rb cd**” and “**cd rb**”. Given these two choices, we pick the sequence containing the tag holding the largest amount of bits, group them into one constituent and update the mutual information values:



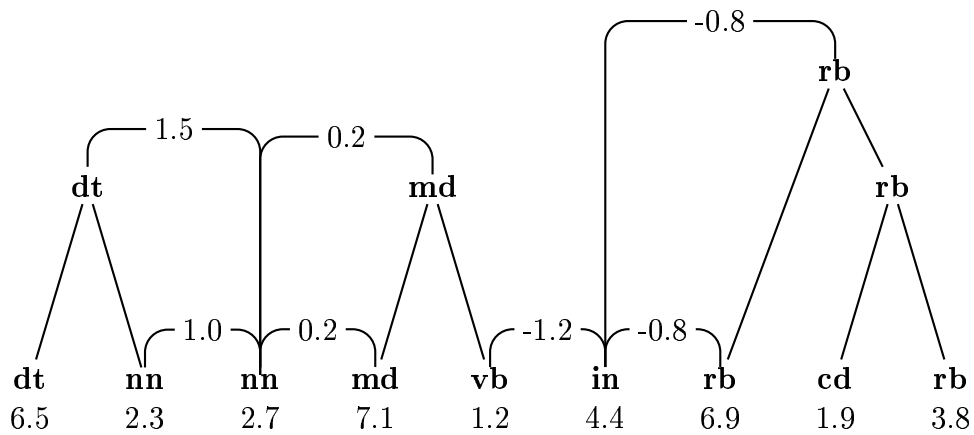
A new link is being made for the “**rb rb**” sequence that holds a rather large mutual information value, but the lexical attraction between “**cd rb**” is higher. This means that the last **rb** element should preferably be attached on the level of the **cd** tag directly and not to the superordinate structure. Doing so, however, causes the following problematic situation:



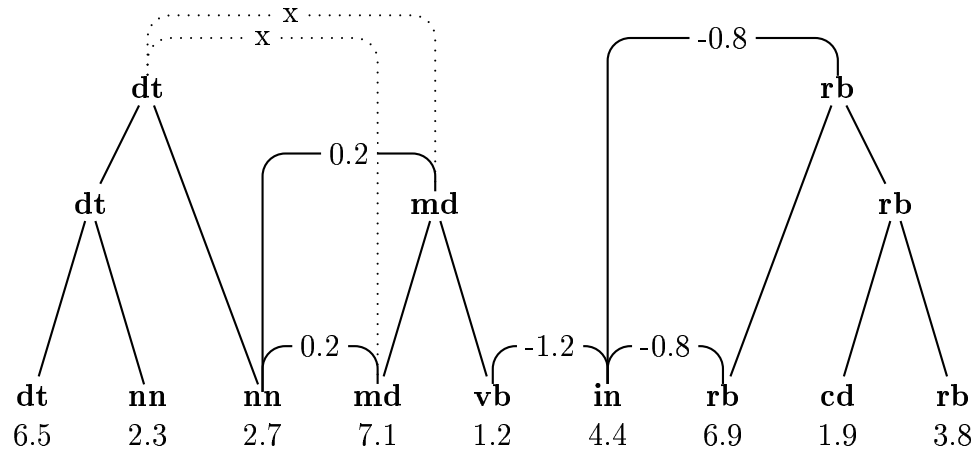
Basic PS-grammar restrictions stipulate that we cannot allow the same terminal to be headed by two different nodes. We therefore attach the newly created superordinate **rb**-node at the node where previously **cd** was attached. We will stipulate later that this operation is only allowed if it does not change the label of the root node of the structure it attaches to. Since this is not the case in the current situation, we can update the structure as follows:



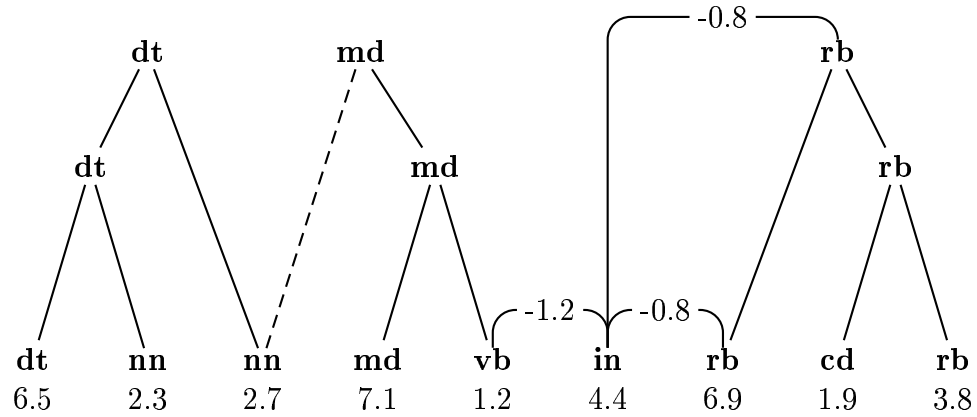
The next bigram we join is “**dt nn**”:



Next up is the newly created “**dt nn**” sequence whose lexical attraction (1.5bits) is higher than that of the other bigrams:

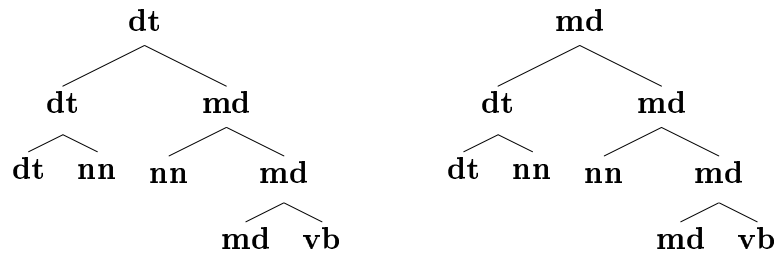


This leaves us with an interesting situation: there are now two equally likely attachment possibilities for the second **nn**: we can directly attach it to the following tag **md** or its superordinate node. Since they both carry the same label, the lexical attraction between the two is the same. Given the choice between levels of attachment, we choose the highest, since the node at that level can be considered to incorporate information content values of its subordinates⁶. This would bring us to the following state:



⁶Also note that the flattening operation described later, often ignores the level of attachment committed to during the building of the tree.

But this again violates the constraint that one terminal node cannot be directly headed by two head nodes. If we were to follow the same procedure we used to resolve the issue with the “**rb cd rb**”-sequence, we would create the left substructure below, the root node of which is replaced by that of the terminal carrying the highest entropy value in the constituent, yielding the structure on the right:

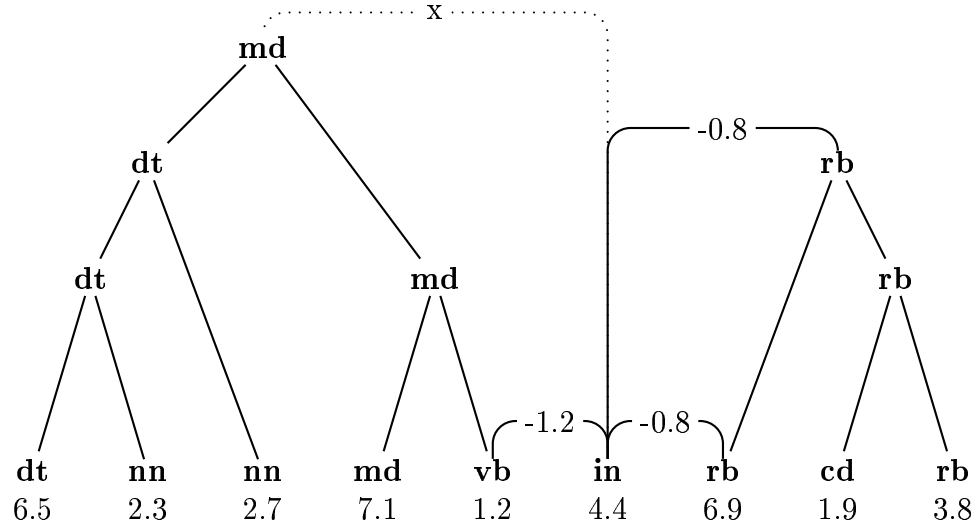


This is of course a highly undesirable structure, but fortunately one our grammar induction method rejects. But why? Attaching the **md**-constituent on a leaf node of the **dt**-constituent is counterintuitive in that it replaces a node carrying 2.7 bits to accommodate a structure whose root node, i.e. the element with the highest information content, carries 7.1 bits. We committed to using the tags with the highest information content as the label for our superordinate nodes on the basis of the intuition that they should be featured in higher-order structures. Attaching it at the level of a node carrying a considerably less amount of bits is therefore counterintuitive, unless there is a very strong lexical attraction warranting this kind of operation (as was the case for the “**cd rb**” sequence).

This would pose not much of a problem if it did not override segmentation and labeling properties earlier committed to by the induction method. But in this situation, the attachment of the **md**-constituent changes the original label of the **dt nn nn** sequence. This is undesirable, since there was obviously enough lexical attraction to commit to this label at an earlier point than the current state. We therefore stipulate that a node can only be attached at a leaf-node of a constituent, if the root-node of the structure to which the constituent is attached can maintain his node label, i.e. if it carries more bits than the node to be attached.

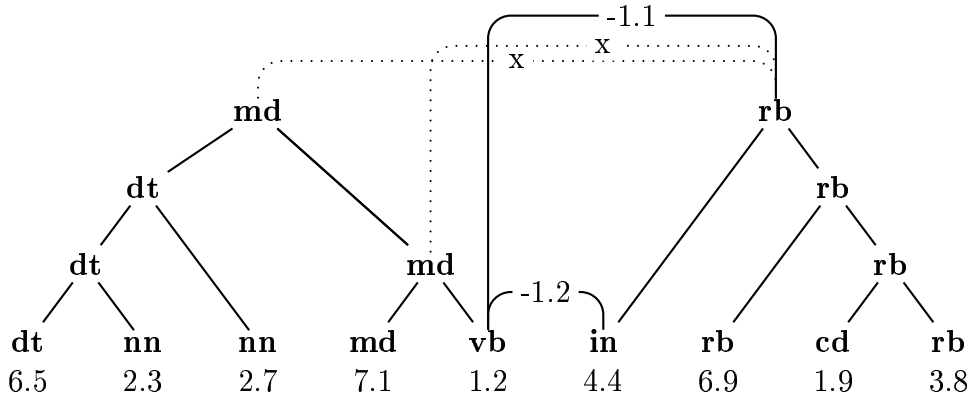
If this is not the case, we climb higher in the structure of the place of

attachment and look at its superordinate node. Unless we can find some alternative node on our path to attach the constituent to, we create a new root node for both root nodes as follows, despite the earlier observed lack of lexical attraction between the **dt** and **md** nodes:

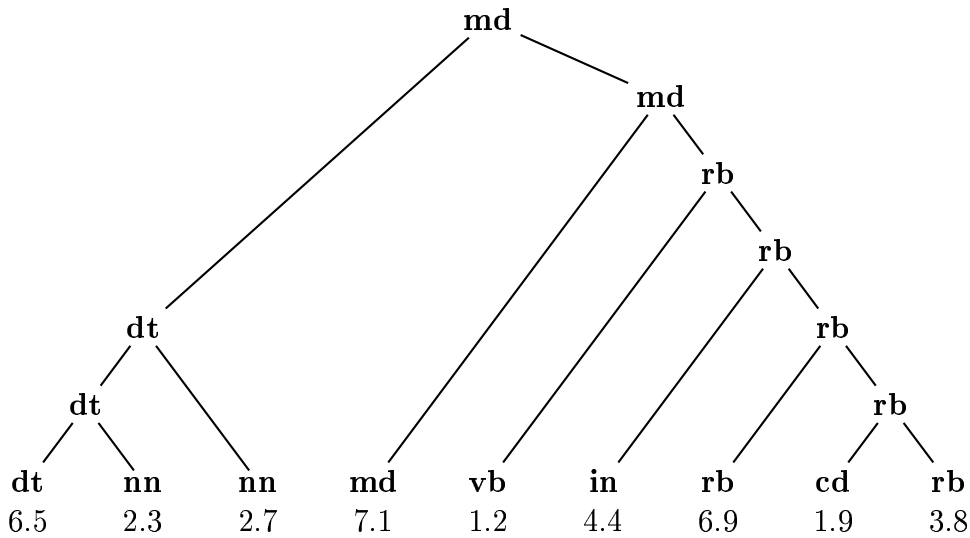


This structure does not change the labeling and segmentation properties of the “**dt nn nn**” sequence, which the induction method had committed to earlier on on the basis of its mutual information.

Sequences with negative mutual information content values were not considered by [Yuret 1998]. But since the grammar induction method we describe here searches for a syntactic structure in a greedy fashion, it considers every kind of lexical attraction starting from the strongest and ranging to the lowest. The current sequence under consideration is “**in rb**” for which a new structure is created:



With only one link to revolve, we are again faced with the aforementioned problem: the **vb**-node is already headed by the **md**-node. Attaching the **rb**-node on the leaf node currently containing **vb** would involve replacing a low entropy node with a high entropy one. We have stipulated that a structure may only be attached to a leaf node, if it does not involve changing the root-node label. It is a close call between the information content of **rb**(6.9) and **md**(7.1), but attachment is allowed yielding the following final structure:



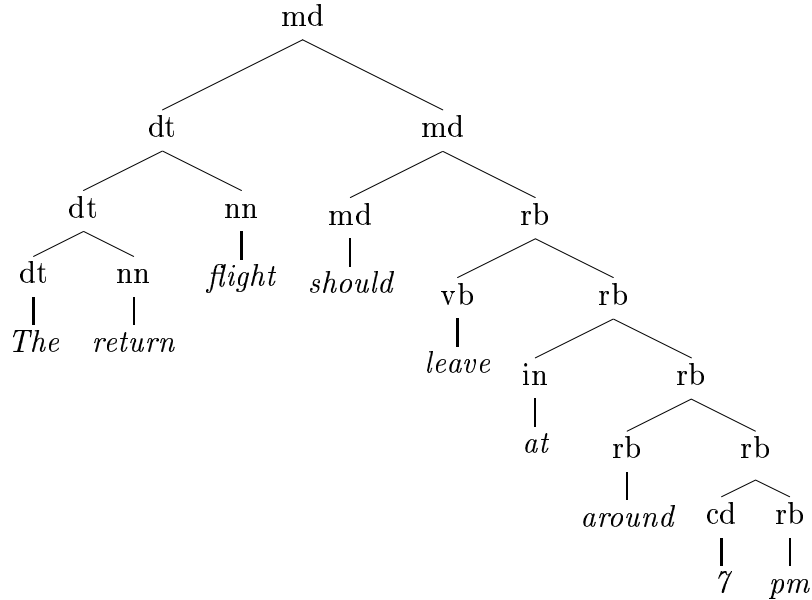


Figure 8.3: Binary Branching output

We can now compare the original tree in Figure 8.2 (p. 269) with the one found by our grammar induction method in Figure 8.3. It is clear that these trees apply a form of segmentation that is to a certain extent similar. The difference between the trees can mainly be explained by the binary-branching structure that the grammar induction method outputs. We can consider flattening tree-structures to make them more similar to the ones found in the ATIS corpus. We can achieve this by compacting a branch with nodes carrying the same label. This flattened tree can be found in Figure 8.4. This structure does retrieve the correct segmentation for “*The return flight*”, but loses a lot of structural information in the rest of the tree.

But we can also process the binary-branching tree by relabeling the nodes: if only one of the two branches in a node is a terminal, we percolate the label of the terminal to its direct head. If the node’s branches expands to two terminals or two non-terminals, the node receives the label of the symbol with the highest information content. Processing the tree like this produces the re-labeled tree in Figure 8.5.

If we flatten our tree using the same principle as applied in Figure 8.4,

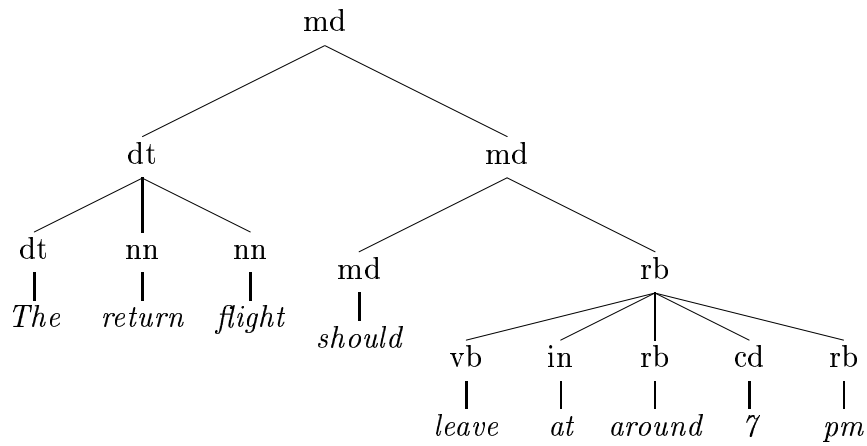


Figure 8.4: Flattened Tree

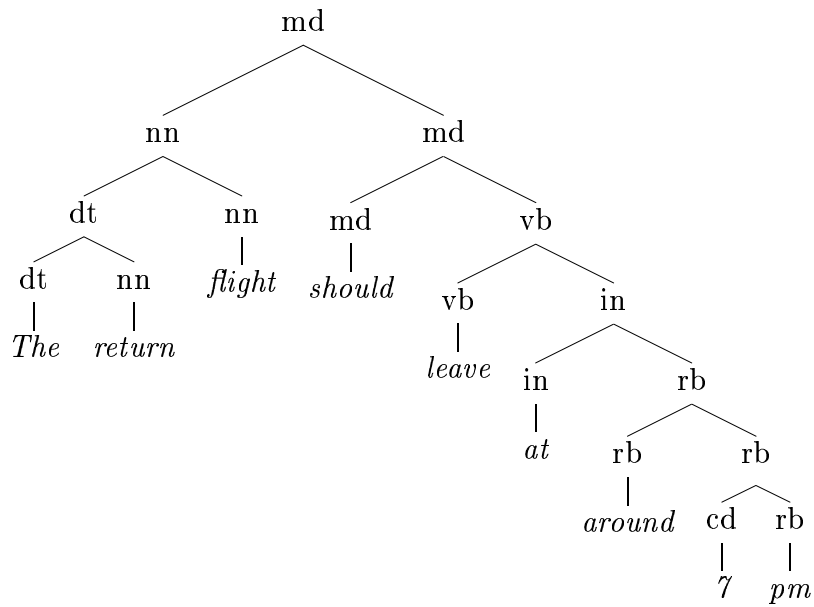


Figure 8.5: Relabeled Tree

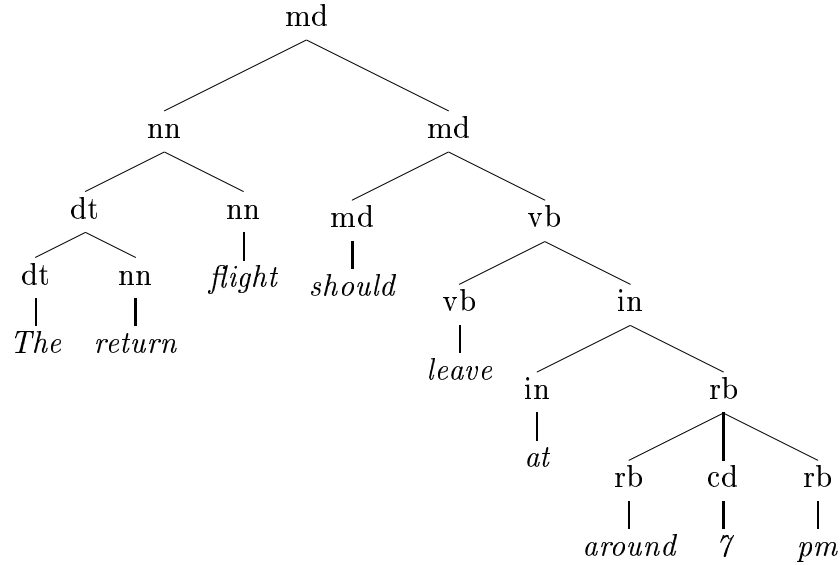


Figure 8.6: Relabeled and Flattened Tree

we obtain the tree in Figure 8.6. Even though this produces an unnecessary embedding in the “*The return flight*” sequence, it does retrieve the correct segmentation for the rest of the sentence. Also note that the labels attributed to the nodes are on the whole close in spirit to those featured in the ATIS-representation of Figure 8.2.

This section introduced a grammar induction method based on simple bi-gram probabilities. Whereas previous methods employed a rigid application of maximum likelihood estimation to construct the parse tree that minimizes the entropy contained in it, this induction method uses a greedy algorithm that builds trees on the basis of the affinity, i.e. lexical attraction, between subsequent words.

Using actual values induced from the ATIS corpus, we were able to find a tree-structure that very closely matched the original gold-standard of the annotated corpus. Unfortunately, the grammar induction method does not

work equally well on all sentences alike⁷. Test results will show that it is vulnerable to the distribution of the data and that small local differences in mutual information content can produce dire parses globally. This means that our unsupervised grammar induction method⁸ will yield a highly defective grammar altogether. However, chapters 5 and 7 introduced an agent-based system that goes a long way in resolving these kinds of limitations in grammars.

8.2.3 Bootstrapping GRAEL-3

The basic idea of the GRAEL system is maintained in GRAEL-3: knowledge is distributed over a society of agents who will adapt to each other in language games and optimize it for inter-agent communication. Whereas in GRAEL-1 and GRAEL-2 the knowledge distributed over the agents took the form of syntactic tree-structures, this kind of information is not readily available at the onset of the GRAEL-3 society. The grammar induction method we described in the previous section (henceforth GIM) can be used to bootstrap structural knowledge in the GRAEL-3 society in several ways.

The first method which we will dub GRAEL-3A is displayed in Figure 8.7: the bare unannotated sentences are run through the GIM which produces an annotated set of tree-structures, which can be distributed over the agents. This method most closely resembles previously described GRAEL-systems. But we can also use the GRAEL-3B method, described in Figure 8.8. Here the raw data itself is distributed over the agents, each of which will apply the GIM on its own data.

Having defined two methods to provide the agents with knowledge, there is still one issue to be resolved: how exactly is knowledge shared in the context of a language game. And more importantly how are language games played? We now have two methods for creating a structure: a parser using the induced ps-grammar⁹ to create analyses (cf. GRAEL-1 and GRAEL-2) and the grammar induction method itself. As GRAEL-3 implements interaction

⁷An example of how the grammar induction method becomes pigeon-holed is described in Appendix G.

⁸and in fact any grammar induction method.

⁹As usual it is extended to incorporate memory-based aspects (PMPG).

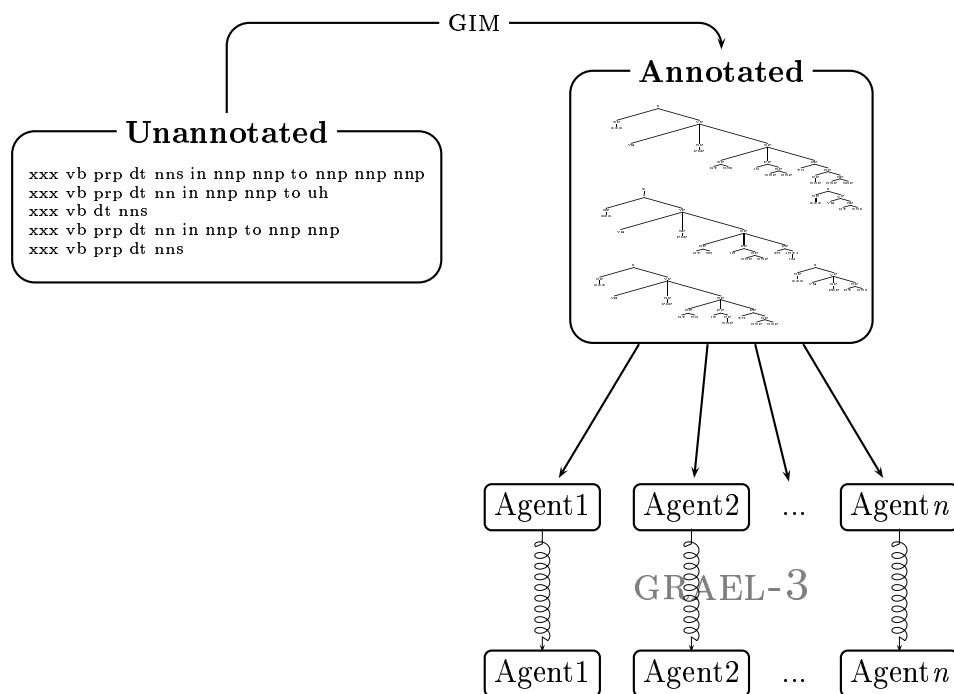


Figure 8.7: GRAEL-3A

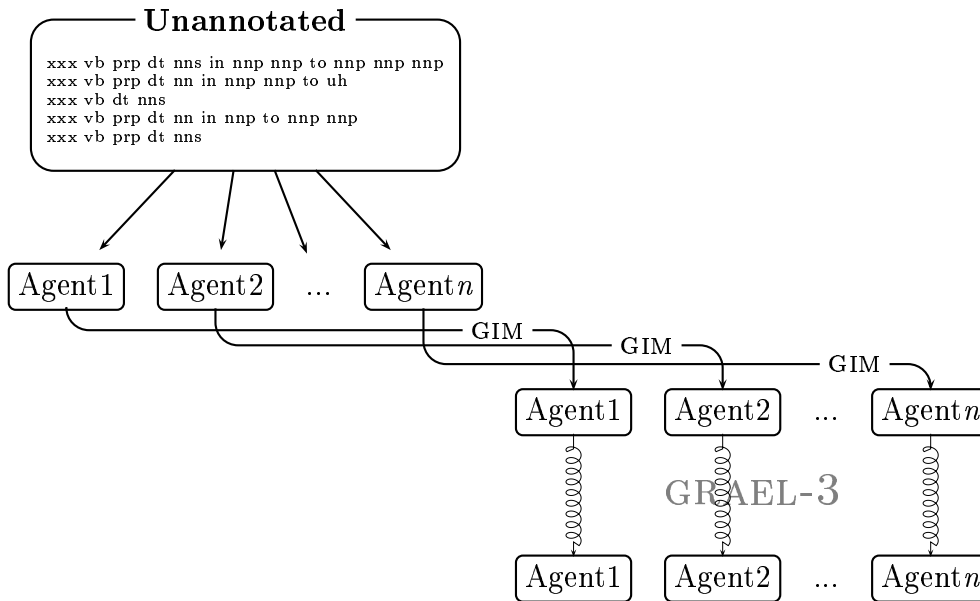


Figure 8.8: GRAEL-3B

between the I-language and the E-language, there are three points at which parsing occurs: (1) an agent parses other agents' sentences in a language game, (2) an agent provides new structures for his E-language using newly acquired information and (3) the fittest agent in a society parses the test set¹⁰.

Table 8.1 hints that there is a wide range of possible combinations, which we will not exhaust in this chapter. We will only consider the two GRAEL-3 instantiations described in Table 8.1. As a point of reference, we include GRAEL-1 in this table, which used PMPG¹¹ for all occurrences of parsing. GRAEL-3-1 bears a close resemblance to GRAEL-1, except that it starts off with a corpus annotated by the GIM.

GRAEL-3-2 not only uses GIM to provide initial structures for the sentences, but also to conduct parsing in the language games as well. Language games are still being played in the same way as before but with some slight differences: **agent1** proposes a parse for a sentence of **agent2**'s E-language.

¹⁰We will not require a validation set in the GRAEL-3 experiments.

¹¹Actually short for the combined system PCFG+PMPG.

	Grammar Induction	Language Games	E-language	Test Set
GRAEL-1	PMPG	PMPG	PMPG	PMPG
GRAEL-3-1	GIM	PMPG	PMPG	PMPG
GRAEL-3-2	GIM	GIM	GIM	PMPG

Table 8.1: 2 instantiations of GRAEL-3

agent2 then compares the structure provided by **agent1** to his own and finds the minimally correct substructure needed for correct parsing. But instead of sharing the structure, **agent2** repeats the terminal nodes of the minimally correct substructure to **agent1**. When the society is halted, the fittest agent can use the GIM to construct a parse directly on the basis of the I-language he compiled over the course of the language games.

Note that, coupled with the previously described distinction between GRAEL-3A and GRAEL-3B, there are in principle four different GRAEL-3 instantiations to be experimented on: GRAEL-3A-1 and GRAEL-3A-2 on the one hand and GRAEL-3B-1 and GRAEL-3B-2 on the other. Note however that from an experimental point of view, GRAEL-3A-2 and GRAEL-3B-2 are identical, because there are no actual structures being distributed at the onset of the society, which dissolves the distinction between GRAEL-3A-2 and GRAEL-3B-2. Let us now turn to the discussion of the experiments.

8.3 Experiments

We have already mentioned the problematic evaluation of grammar induction methods in Chapter 7 and Section 8.1. An interesting proposal was made by [van Zaanen and Adriaans 2001] to evaluate unsupervised grammar induction methods in the same way as we would a supervised method: by measuring the accuracy of the method in terms of precision and recall (cf. Chapters 3,5 and 7). [Clark 2001] criticizes this approach by pointing out that the gold-standard structures an annotated corpus provides, are the result of human annotators making arbitrary decisions and that the structures provided in the annotated corpus do not reflect some “*theory-independent reality*”. This point is moot however, since there is no such thing as syntactic structures that reflect a theory-independent reality. [Clark 2001] therefore argues in favor of a qualitative analysis of the grammar induction method rather than

a quantitative, but fortunately continues to provide both. We will employ the same evaluation method in this chapter.

8.3.1 Experimental Setup

We will describe three types of experiments: the first set of experiments featured GRAEL-3 trained and tested on the ATIS-corpus. This restricted domain will allow us to consider a number of different experimental settings, including the three previously defined GRAEL-3 instantiations. The next set of experiments uses the WSJ-corpus as a training set. Both experiments with words and tags will be described, while different GRAEL-3 instantiations are tested on the test sets for the ATIS corpus and Section 23 of the WSJ-corpus. A final batch of experiments provides a limited qualitative analysis, as we apply GRAEL-3 on large amounts of texts from a limited domain.

Grammar induction methods generally need all the data they can get, so one might be inclined to simply use single-epoch GRAEL-3 societies. One of GRAEL-1's strengths as a grammar optimization method however was its ability to weed out bad grammatical information from a society. Given the fact that the GIM indeed creates a lot of erroneous grammatical structures, a generation-based system therefore seems advisable. All GRAEL-3 societies described here are crossover-based societies, using a 10-agent society using understanding accuracy information to determine agent fitness. Each agent slot is given 10 generations¹² and the experiment was stopped after the last agent disappears.

Note that in a GRAEL3-2 type society, grammatical knowledge is not represented as actual structures, but is induced from a collection of sequences of part-of-speech tags. In a typical GRAEL-society newborn agents are created by joining two agents' top 50% most probable rules and a selection of the other structures. This kind of probabilistic information is of course not present in an I-language solely consisting of part-of-speech tag sequences. The sequences for newborn agents are therefore randomly selected from the ancestors' I-languages.

Noisy channel mutation was not implemented in the GRAEL-3 experi-

¹²Generation shifts were triggered using the formula on p. 183.

ments, as the development of the grammars is already unstable. Since we need to take into account that the structures in the initial E-languages are subpar and can therefore a priori not function as a gold-standard touchstone for the agents, we do allow the agents to re-parse their E-language after every language game run.

8.3.2 Restricted Domain - ATIS

Let us first consider GRAEL-3A-1 applied on part-of-speech tags as it is the closest in spirit to the previously described GRAEL systems. To recap: GRAEL-3-1 annotates a collection of sentences (sequences of part-of-speech tags) using the GIM. The trees are re-labeled, but not flattened¹³ (cf Figure 8.5) and distributed over a society of 10 agents. The agents use these initial structures to induce a PMPG which is consequently expanded and optimized in a series of language games using a GRAEL-1 type approach.

We use the same data set partitioning as in the GRAEL-1-experiments: a 58 sentence test set was used to evaluate GRAEL-3, while the remaining 463 sentences, together with the unused validation set, was used as a training set. As a first step, the GIM provided structures for the 520 sentences in the training set. We then induce a PMPG from these structures and record its precision and recall. Since there is no straightforward way to match the node labels provided by GIM to those featured in the original corpus, we measure **unlabeled** precision and recall and look at the number of correct constituents, regardless of their category label. The grammar obtained from the GIM-structures achieves a **22.4%** F-score on the test set.

Following [van Zaanen and Adriaans 2001; Clark 2001], we also adopt the somewhat looser criterion **OCB** (zero crossing brackets). This measures the amount of times the brackets of a parse¹⁴ crosses those of the gold-standard. Note however that the OCB measure tends to overestimate the importance of flat parses, as they are less likely to have crossing brackets. Baseline OCB-accuracy for the initial grammar is 21.3%. If we use the grammar to parse the training set, we get higher scores, but still considerably lower than those obtained from a supervised induced grammar: **22.7%** F-score and 24.3%

¹³The flattening operation would eliminate recursion in our grammar.

¹⁴The brackets of a parse are identical to the constituent boundaries.

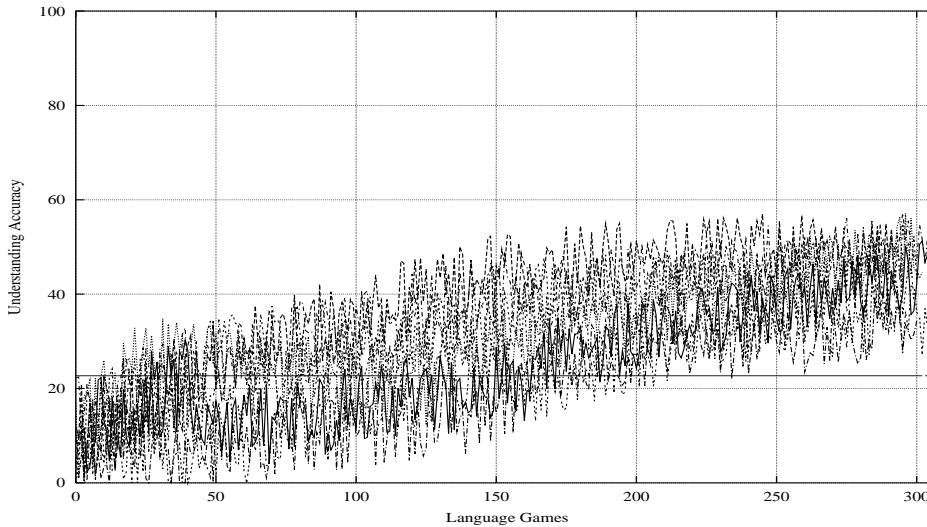


Figure 8.9: GRAEL-3A-1 - ATIS - Understanding Accuracies

OCB. This indicates that the grammar found is far from optimal in that it is not able to generalize over the data in such a way as to make accurate parsing possible. The main problem lies in the labeling which hampers the usability of the ps-grammar in a parser.

To see if our GIM is actually doing better than the lower threshold baseline accuracy we can expect, we also conducted an experiment in which we randomly generated tree-structures for sentences. We achieved a 6.7% F-score for the sentences on our training set and a **7.8%** F-score for the sentences of our test set. Using these structures in a GRAEL-3-A-1 society did increase the scores up to 8.2%, but the difference was not significant.

Figure 8.9 shows the understanding accuracy plots for this experiment. The baseline accuracy is set at 22.7%, as this is the accuracy achieved by the grammar induced from and tested on the training set. The society starts off with very low understanding accuracies with many agents being totally incomprehensible to one another. As the agents acquire structures in language games and share grammatical knowledge, they slowly start to reach a state of convergence and agree on a larger number of grammatical structures. They eventually achieve understanding accuracies up to 55%. This only indicates that the agents are converging and does not necessarily mean

that they have acquired a better grammar than the one that was distributed over the society. Understanding accuracy in the last 50 runs of the society circles around 40%.

Note that, even though this is a generation-based society, this does not seem apparent from the plots in Figure 8.9: this is due to (1) the dispersed nature of the plots, concealing accuracy shifts and (2) an apparent stability in parsing behavior, even when part of the grammar is taken away. This may suggest that the agents are mainly using the most probable portion of their grammar for parsing, which is always transferred to the offspring.

Table 8.2 displays the accuracy achieved by the fittest agent, when the society is halted at the 285th language game run (understanding accuracy halting point). As a point of reference we also include accuracy scores for EMILE and ABL [van Zaanen and Adriaans 2001] and CDC [Clark 2001]. The latter however uses a different adaptation of the ATIS corpus than [van Zaanen and Adriaans 2001]. The results on GRAEL-3 report on a different version of the ATIS-corpus¹⁵, making a direct comparison troublesome. We include the figures since they are our only basis of comparison and should provide some general idea of GRAEL-3's performance.

The result of GRAEL-3A-1 improves considerably on the baseline model, but a **25.6%** F-score is still a rather modest result, even for an unsupervised grammar induction method, especially compared to the other methods. Note that for both the baseline method as well as for GRAEL-3A-1, precision is much higher than recall. This can be explained by the poor ps-grammar obtained by both methods: a lot of sentences could not be parsed at all, limiting the total number of constituents that are generated, which affects the precision score as it expresses the ratio of correct constituents vs the total number of constituents generated.

Let us look at GRAEL-3AB-2 which uses GIM consistently for all parsing operations. Figure 8.10 displays the understanding accuracy plots for this experiment. We calculate a new baseline accuracy, by using the training set to induce structures for the test set using the GIM. Surprisingly perhaps, this yields a significantly higher score than the PMPG-approach. The baseline accuracy on the training set for the GIM is 33.1%. The first thing we notice when we look at the results for GRAEL-3AB-2 in Table 8.2 is that precision is

¹⁵ATISII holds less trees and on the whole less homogeneous trees.

	UP	UR	$F_{\beta=1}$	OCB
EMILE	16.8	51.6	25.4	47.4
ABL	35.6	43.6	39.2	29.1
CDC	34.6	53.4	42.0	45.3
Baseline (PMPG)	25.1	20.2	22.4	22.9
GRAEL-3A-1	28.5	23.2	25.6	24.9
Baseline (GIM)	27.4	29.4	28.4	30.8
GRAEL-3AB-2	30.1	31.9	31.0	31.1

Table 8.2: GRAEL-3A-1 vs GRAEL-3AB-2 - ATIS - Results

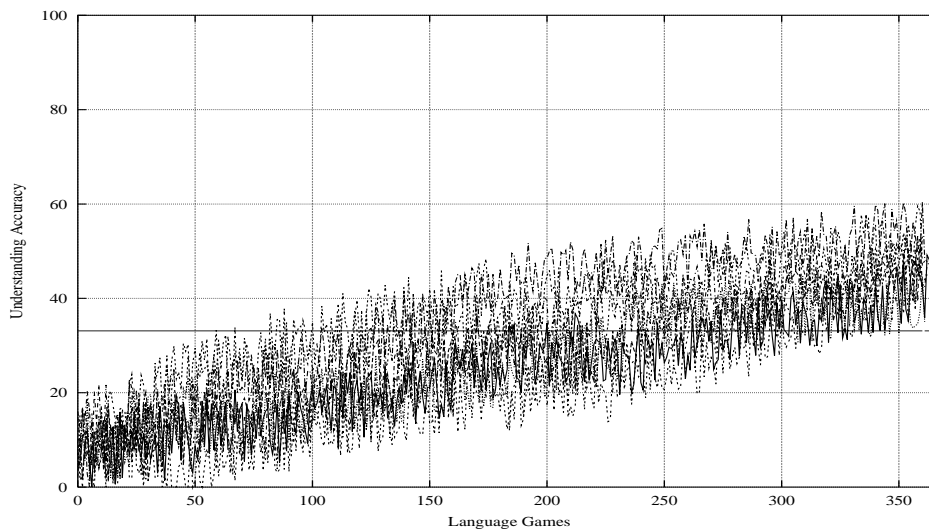


Figure 8.10: GRAEL-3AB-2 - ATIS - Understanding Accuracies

now lower than recall, because the GIM induces a structure for all sentences. This means that on average more correct constituents are generated, but at a lower ratio to the total number of constituents created. The overall F-score for the fittest agent in the GRAEL-3AB-2 society is 31.0%, a significant increase, both over the baseline method and GRAEL-3A-1.

We now turn to the GRAEL-3B instantiations which provide the agents with part-of-speech tag sequences at the onset of the society, rather than full tree-structures. The agents then apply a GIM to those sentences and end-up with an annotated set of trees. But whereas the structures in GRAEL-3A were based on distributional information about the entire training set containing 520 sentences, the initial grammars in GRAEL-3B are each based on only 1/10 of that amount. The GIM is a greedy algorithm and will come up with structures regardless of the size of the training set, but the quality of those grammars seems a priori questionable.

If we interpret the agents in GRAEL-3B as bag-like entities, we might however conjecture that among all the grammatical structures that exist in the society, many are erroneous, but some might also benefit from some lucky sampling of the original training set and constitute useful grammatical structures. The GRAEL society should then try to distinguish erroneous from useful structures. Note that from an experimental point of view GRAEL-3A-2 and GRAEL-3B-2 are identical, since there are no actual structures being distributed at the onset of the society.

Figure 8.11 shows the course of the GRAEL-3B-1 experiment. Optimizing the grammar in this environment is apparently very problematic, with a very gradual learning curve. This time, newborn agents seem more affected by the newly compiled grammar, as minimal F-scores can be observed until as far as the 200th run. The society gradually picks up after that, but is still only barely able to outperform the baseline model. The fittest agent in this GRAEL society achieves an F-score of **22.7%** which is comparable to that of the baseline (Table 8.3). Note how precision is again much higher than recall. In fact, the recall score for GRAEL-3B-1 is much lower than that of GRAEL-3AB-2. The disappointing scores for this GRAEL-3 instantiation shows that it is not very well suited to this kind of optimization task.

Apparently the GRAEL-3-2 method of using GIM for all parsing tasks, is more capable of optimizing the initial grammars. This is not surprising: we

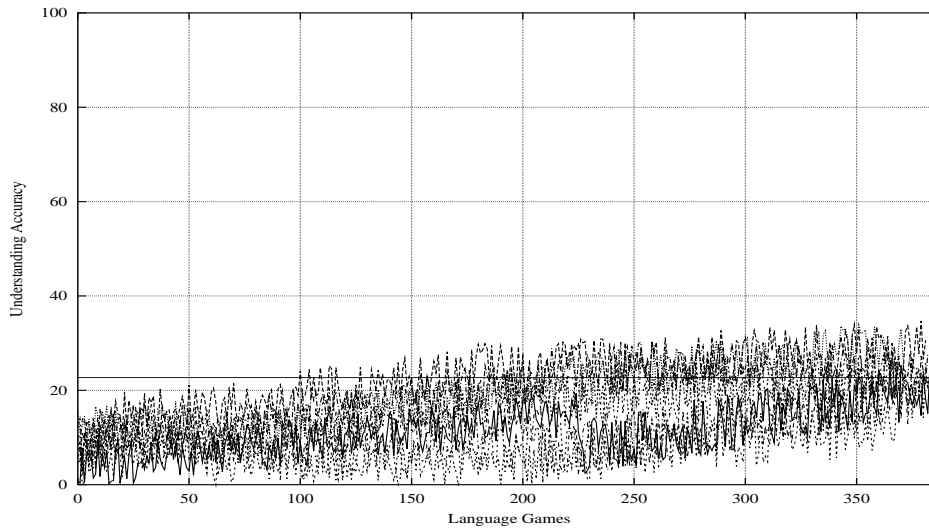


Figure 8.11: GRAEL-3B-1 - ATIS - Understanding Accuracies

	UP	UR	$F_{\beta=1}$	OCB
Baseline (PMPG)	25.1	20.2	22.4	22.9
GRAEL-3A-1	28.5	23.2	25.6	24.9
GRAEL-3B-1	26.6	19.8	22.7	22.9
Baseline (GIM)	27.4	29.4	28.4	30.8
GRAEL-3AB-2	30.1	31.9	31.0	31.1

Table 8.3: GRAEL-3A-1 vs GRAEL-3B-1 vs GRAEL-3AB-2 - ATIS - Results

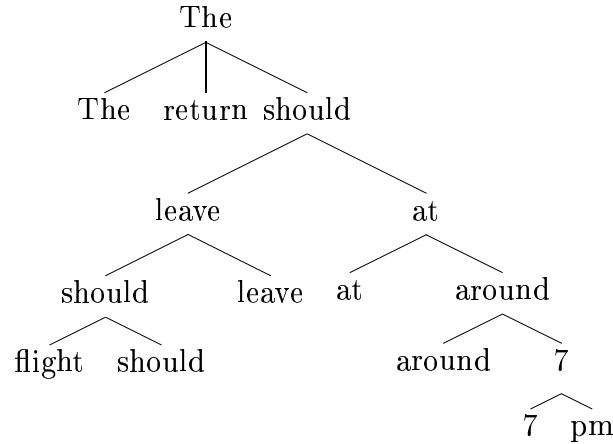


Figure 8.12: Word-based Parse

identified the labeling as one of the main problems in parsing with a PMPG-grammar based on structures induced by the GIM. If these structures are based on a tiny dataset, the way the structures are built is largely dependent on very local lexical attraction values and is therefore to a large extent guided by chance. The labeling that these structures yield can be expected to be less accurate as well. And since a parser using a ps-type grammar uses these labels to generate structures, it is therefore not surprising the agents do not perform very well at the onset of the society and that even GRAEL cannot salvage the situation over time. In the GRAEL-3B-2 method, structures become gradually better when new information is introduced, so that local maxima are more easily overcome.

Considering the tiny data set that the GIM had to base its grammar on, the results are encouraging. Even though transforming the structures into ps-grammars is not a good idea, GRAEL-3 does seem to improve both the performance of societies wielding phrase structures and of societies powered by the GIM exclusively. But compared to other methods, the overall results are disappointing and it seems that if we are to exploit the grammar optimization capabilities that GRAEL provides to the fullest, we will need a larger data set that can deliver more fine-tuned information content values, that are not as vulnerable to the negative side effects the distributed nature of GRAEL can cause.

Before we turn to a set of experiments that uses a larger dataset, we briefly look at unsupervised grammar induction on the word-level. Due to the size of the ATIS corpus, the GIM was not able to induce a decent grammar for parsing strings of words. To illustrate this, we show the (re-labeled and flattened) structure for the sentence “the return flight should leave at around 7 pm”¹⁶ in Figure 8.12. If we train and test a GIM on the words of the ATIS-corpus, we only achieve a 10.5% F-score. If we want to generate structures on top of the words of the ATIS-corpus, we will need a larger dataset to adequately estimate the (mutual) information content of the words in the corpus.

8.3.3 “Unrestricted” Domain - WSJ

The experiments on the ATIS-corpus showed that our GIM is able to build a grammar that can provide parses that range from impressively accurate (cf. the “*the return flight...*” sentence) to nonsensical (cf. Appendix G). Trained on 520 sentences and tested on 58 similar sequences, the GIM achieved a remarkably high, if still overall underwhelming F-score. Transforming the induced grammar to a ps-grammar has been shown to produce substandard results, mainly due to the aforementioned problem of labeling. A method that pre-distributes the unannotated sentences over the GRAEL-society only amplifies this problem.

The GIM used by GRAEL-3 seems able to provide parses for sentences based on a very limited amount of data. This begs the question: does the performance of GIM benefit from the fact that training was performed on a set of sentences similar to the test set, or is its performance in fact limited by the size of the data. We therefore need to experiment on a larger corpus, which will not only enable us to investigate this matter, but should also enable us to perform some experiments for unsupervised grammar induction on words, rather than part-of-speech tag sequences, since the latter can still be considered as a semi-supervised method of grammar induction.

¹⁶“The” is the most salient word, because it is capitalized and only six sentences in the ATIS corpus start with “The”.

Tags

Due to the computational costs of using GRAEL with a ps-grammar backbone, we limited the experiments to only one with GRAEL-3-B-1 on part-of-speech tag sequences. Since the WSJ-corpus is much larger, the issues that limited the performance of this particular GRAEL-3 instantiation in the ATIS experiment should be resolved.

We will first describe the GRAEL-3-B-1 experiment. Like in the previous experiment with the full WSJ-corpus, we first divide the training set (Sections 2 to 21 of the WSJ-corpus) into 40 parts of about 1000 sentences each. Actually, it is divided into 40 parts of 10x100 sentences, with a random partitioning for the 10 agents¹⁷ already present. At the onset of the society, the first partition of 10x100 sentences is distributed over the society of 10 agents. Each agent holds about 100 sentences in his E-language. Contrary to previous experiments the agents now use information on all 40x100 sentences that are attributed to them to power the GIM. This produces tree-structures for their initial sentences.

Next, processing continues similarly to a regular GRAEL-experiment with the agents parsing other agents' sentences using the ps-grammar induced from the structures provided by the GIM and the grammatical structures acquired during language games. In this experiment, however, the sentences in the E-language are not re-parsed after each language game run, as this would be too computationally expensive. Therefore, interaction between I-language and E-language is limited to parsing the sentences of the new E-language that is provided every 40 runs.

Figure 8.13 displays the course of the experiment. The baseline accuracy (26.3%) provided in this plot is the F-score on the 1st partition of 1000 sentences, achieved by the grammar extracted from the structures provided by the GIM from the entire training set. In this graph, we notice at regular intervals some upward, as well as downward peaks when a new partition is introduced in the society, most notably at run 40. Overall, this does not seem to influence the agents' development at all. Although an understanding accuracy halting point can be determined after run 373, the plots seem to

¹⁷As opposed to previous experiments with GRAEL on the WSJ we use a limited population size, to provide each agent with sufficient data to induce an initial grammar.

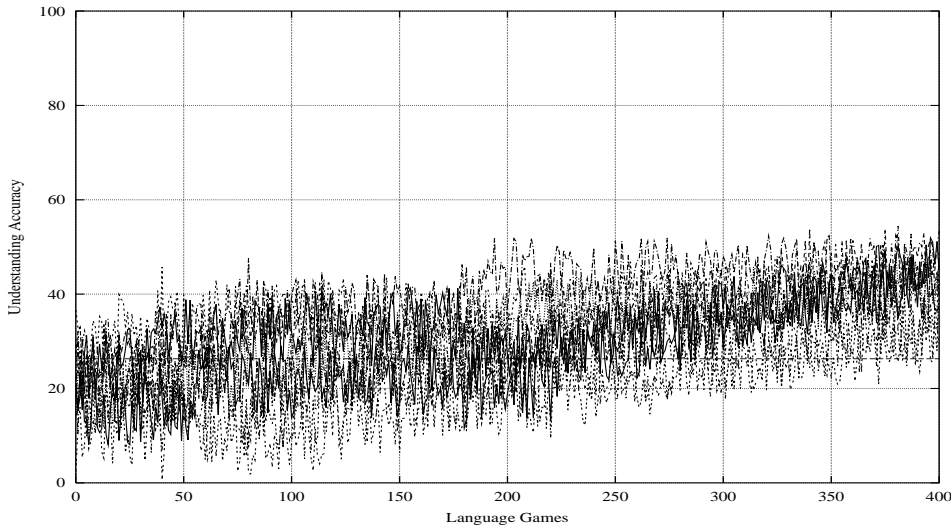


Figure 8.13: GRAEL-3B-1 - WSJ - Understanding Accuracies

suggest that understanding accuracies might rise if the society is allowed to continue. Future research should look into this issue.

When the society is halted, the fittest agent is selected and its grammar is used to parse the test sets (the 58 sentence ATIS test set and Section 23 of the WSJ corpus). The results are encouraging (Table 8.4): GRAEL-3B-1 achieves a **25.3%** F-score on the WSJ test set and a **31.8%** F-score on the ATIS test set. This compares favorably to the GRAEL-3B-1 counterpart trained on the ATIS corpus. The GRAEL-3B-1 seems to scale well to larger corpora, achieving an important performance boost. It outperforms the GRAEL-3AB-2 society, but not by a great margin, especially considering the large amount of extra computation involved. Scores are considerably lower than those reported by [van Zaanen and Adriaans 2001] and [Clark 2001], but this can also be attributed to the increased amount of training data used in these compared to the experiments described here.

Note that for both the baseline model and GRAEL-3B-1 recall is higher than precision when parsing the WSJ test set, but about the same when parsing the ATIS test set. This indicates that many sentences of the WSJ test set could not be parsed, in contrast to ATIS, for which almost all sentences were attributed some parse.

	Test Set = WSJ				Test Set = ATIS			
	UP	UR	$F_{\beta=1}$	OCB	UP	UR	$F_{\beta=1}$	OCB
GRAEL-3B-1 ATIS	—	—	—	—	26.6	19.8	22.7	22.9
Baseline (PMPG)	38.0	16.2	22.8	24.3	25.6	24.8	25.2	27.3
GRAEL-3B-1 WSJ	38.1	19.0	25.3	28.2	31.6	32.1	31.8	32.8

Table 8.4: GRAEL-3B-1 ATIS vs GRAEL-3B-1 WSJ - Results

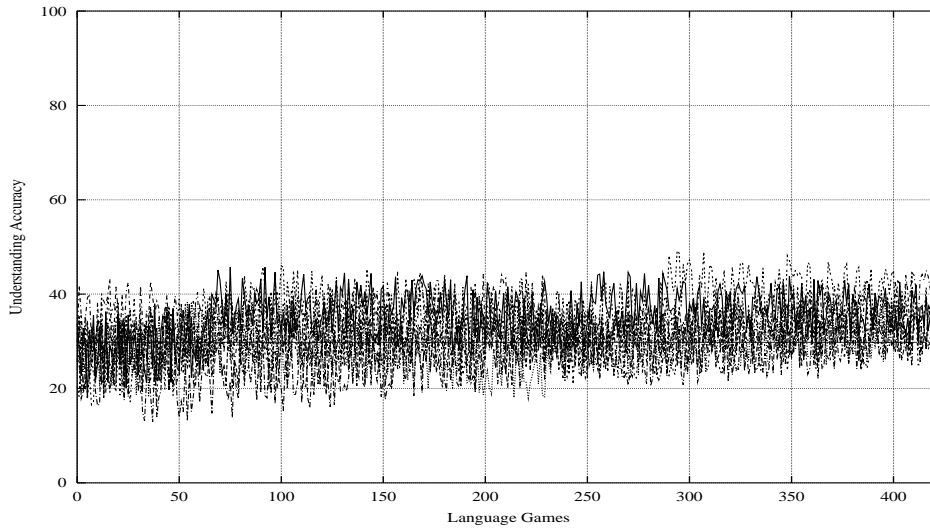


Figure 8.14: GRAEL-3AB-2 - WSJ - Understanding Accuracies

Generating structures with the GIM is a very inexpensive process, since it is a bottom-up, best first-search algorithm. For the WSJ experiment with the GRAEL-3AB-2 method, we can therefore abandon some of the restrictions we normally apply when processing the WSJ dataset in a GRAEL society. In this experiment, all sentences are distributed over the agents in the society at the onset. To still allow the society to move along at a reasonable pace, 100 full sentences are sampled from the agents’ I-language after each language game run to compile an E-language.

In this experiment, we allowed each agent’s slot to create 20 newborn agents, as opposed to 10. Figure 8.14 shows that this is to no avail, as there is hardly any noticeable increase in understanding accuracy among the agents. A very gradual increase can be observed, but overall, the society

	Test Set = WSJ				Test Set = ATIS			
	UP	UR	$F_{\beta=1}$	OCB	UP	UR	$F_{\beta=1}$	OCB
GRAEL-3AB-2 ATIS	—	—	—	—	30.1	31.9	31.0	31.1
GRAEL-3B-1 WSJ	38.1	19.0	25.3	28.2	31.6	32.1	31.8	32.8
Baseline (GIM)	24.9	26.1	25.5	26.6	31.4	33.1	32.2	32.5
GRAEL-3AB-2 WSJ 175	25.2	26.3	25.7	26.6	31.5	33.3	32.4	32.5
GRAEL-3AB-2 WSJ 350	25.2	26.3	25.8	26.6	32.8	34.1	33.4	33.6
GRAEL-3AB-2 WSJ' 350	26.0	26.9	26.5	26.7	33.2	34.5	33.8	34.0

Table 8.5: GRAEL-3AB-2 trained on WSJ - Results

evolves very slowly. So much so that the understanding halting procedure triggers a halting point after run 175. We therefore selected another fittest agent at the arbitrarily chosen halting point of run 350.

Table 8.5 provides an overview of the results. The first line shows the results of the GRAEL-3AB-2 society trained on the test set. This society achieved a reasonable F-score of **31.0%**. It is outperformed by the GRAEL-3-B-1 society that was trained on the WSJ corpus (2nd line). The baseline for this experiment was the GIM applied on the test set, using distributional information from the training set.

The fittest agent at run 175 (GRAEL-3AB-2 WSJ 175 in Table 8.5) increases the score slightly on the WSJ and ATIS test set compared to baseline accuracy. The fittest agent almost 200 runs later (GRAEL-3AB-2 WSJ 350) does yield some performance increase on the ATIS test set, but a hardly noticeable effect on the WSJ test set. In fact, looking at the ATIS test set, there is a difference of less than 10 constituents between the GRAEL-3AB-2 systems described in Table 8.5, which indicates that the beneficial effect of GRAEL-3 is minimal.

Apparently GRAEL-3 does not provide much of a performance gain for this kind of parser. This is not surprising if we consider what is going on in the language games: **agent1** provides a structure for one of **agent2**'s sentences. **agent2** will seek out the minimal correct substructure and provide it to **agent1**. But since **agent1** does not use these structures itself for parsing, only the part-of-speech tag sequence is recorded. However, adding this sequence to an already large I-language, does not have a directly significant

	Test Set = WSJ				Test Set = ATIS			
	UP	UR	$F_{\beta=1}$	OCB	UP	UR	$F_{\beta=1}$	OCB
Baseline	18.1	19.0	18.6	19.0	23.5	23.8	23.6	23.9
GRAEL-3AB-2	20.0	20.4	20.2	20.8	25.2	24.8	25.0	26.2

Table 8.6: GRAEL-3AB-2 trained on WSJ - Words - Results

impact on the agent's parsing behavior. Therefore learning is slow and the increase achieved from GRAEL is minimal.

A bigger performance increase can be achieved if we gather all I-languages of all agents and use the distributional properties of this set of sentences to parse the test sets. This approach (GRAEL-3AB-2 WSJ' 350 in Table 8.5) outperforms any other system discussed so far: apparently the distributional properties observed across the society have been optimized, providing an extra increase for the accuracy scores of the GIM on the test set.

Words

The experiments so far have all been conducted on part-of-speech tag sequences. Ideally, however, we want to be able to induce grammars without the help of human annotators. We therefore present one experiment in which we tried to induce a grammar based on the distributional properties of the words of the WSJ-corpus rather than their tags. A preliminary experiment on the ATIS-corpus had shown that it was too small to extract meaningful structures (cf. Figure 8.12). It may be interesting to see if we can improve the performance of the GIM on words if we have a larger data set at our disposal.

We trained and tested a GRAEL-3-AB-2 society on the words of the WSJ training set and tested its fittest agent on the ATIS test set, as well as on Section 23 of the WSJ corpus. The results are not very good (Table 8.6): there is a significant drop in accuracy on the WSJ test set, although not as great as for the ATIS test set. The GIM is having a hard time extracting distributional information for words. Especially the ATIS corpus does not seem to be easily parsable on a word-level.

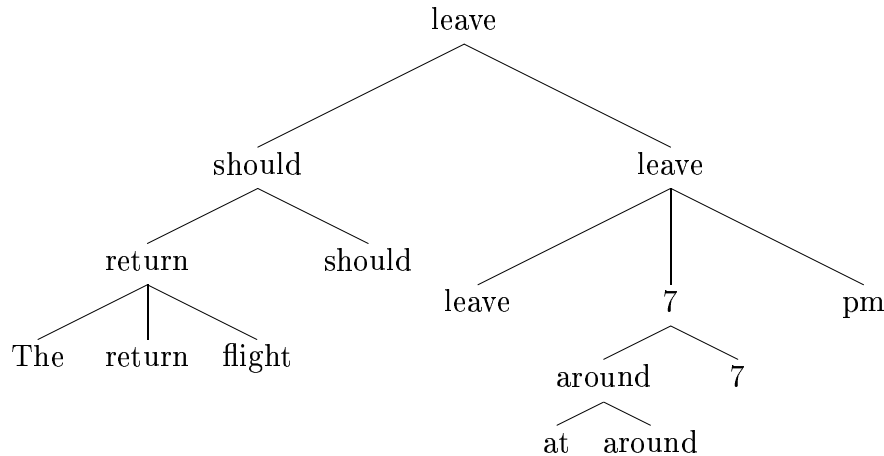
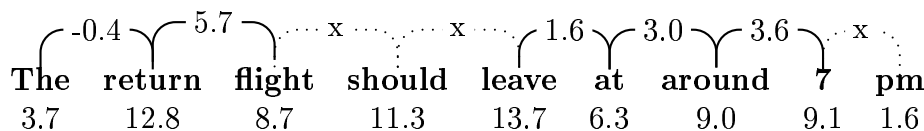


Figure 8.15: Word-Based Parse

Let us turn to our standard example to see what is going on. Here is the sentence to which the mutual information content values have been added¹⁸:



First thing we notice is that now the item carrying the largest information content is the actual verb, which is a better estimation than previously suggested (**md** and **The**). But there are two very important arcs missing: the one connecting **7** and **pm**, **leave** and **at**, and most importantly, the one connecting **should** and **leave**. Upon inspection of the data, it is indeed the case that there is no occurrence of the bigram “should leave” in the WSJ-corpus. As both elements carry a high number of bits, they each occur at the head of each other’s half of the sentence, which makes it practically impossible to attach the two structures properly. The final structure proposed by the GIM can be found in Figure 8.15, although not completely without merit, is obviously a far cry from the one we obtained in Figure 8.6.

¹⁸If there was no mutual information content, the unigram information content of the word was used, rather than the bigram information content.

There are a number of limiting factors at work here: the WSJ-corpus provides a way to consider unsupervised grammar induction on words, but at 1,056,519 words¹⁹ it is seemingly still not large enough to capture not uncommon phrases such as “should leave” or “7 pm”.

The internet provides an infinite corpus to extract distributional information, so that given enough data, we should eventually be able to capture these kinds of sequences. But this argument misses the point: to capture these sequences, we perhaps need some form of generalization, like part-of-speech tags. It is obvious that a modal followed by an infinitive is a very common sequence and our grammar induction method should know about this rather than wait till it has observed all combinations of modals and infinitives. Unsupervised grammar induction on words should therefore perhaps limit its scope to restricted domains (cf. ATIS), provided there is enough data to capture the most relevant sequences, or should either generalize over the words in the form of part-of-speech tag sequences. Recent research however [van den Bosch and Buccholz 2002] suggests that the latter type of generalization is not required and that a word-based system for shallow parsing can outperform a system that generalizes over the data in the form of part-of-speech tags, provided there is enough data available.

On a more general level, it is also clear that a simple bigram approach to grammar induction is missing a lot of lexical relations between words. An obvious way to increase the performance of the GIM would be to look beyond the range of the surrounding words for grammatical relations. This might provide a better way to build structures on top of sequences of words and tags alike.

8.3.4 Restricted Domain

In this section, we would like to take a quick look at how we can apply GRAEL-3 to a large amount of texts from a restricted domain. This will allow us to test the capabilities of our GIM and GRAEL-3 on a pre-selected text type.

We tried to look for a type of text that does not just feature homoge-

¹⁹Compared to e.g. 15,000,000 words in [Clark 2001].

nized sentences like ATIS does, but which is not unrestricted either in the sense that WSJ is. Our domain should feature a sufficient range of sentences of unrestricted content, but also have a number of sentences featuring some recurring terminology. This will allow us to see how well our grammar induction method is able to make syntactic generalizations on both types of sentences.

The domain we chose was that of DVD-reviews on the internet. Typically, a DVD-review will consist of a discussion of the movie featured on the disc (i.e. unrestricted content), as well as a discussion of the technical features of the disc (recurring terminology). The internet-site **DVD-Basen**²⁰ is a portal site that links to 63063 DVD-reviews on the internet at the time of writing. Using the search term “the”, we downloaded thousands of reviews, stripped formatting information and headings from the text and randomly drew about 250.000 sentences, leaving us roughly with a 5 million words training set.

We distributed these 250k sentences over 10 agents in a GRAEL society. The agents then played a series of 400 language game runs in a single-epoch society. There was no interaction between I-language and E-language and the E-languages consisted of 200 sentences randomly drawn from any of the full sentences in the agents’ I-language. After the 400th run, the society was halted and the agents’ I-languages were compiled into one big training set. This training set was consequently used to power the GIM which provided tree-structures for the text of a DVD-review for the movie “Clerks”²¹.

We distinguish two types of texts in a DVD-review: unrestricted content text, describing the story of the movie and the technical discussion of the disc featuring a recurring terminology. As a qualitative analysis of GRAEL-3, we will discuss the structures induced for sentences of each text type.

First we look at two sentences from the description of the movie:

1. Clerks is the story of two, well, clerks.
2. Writer and Director Kevin Smith has a great knack for writing dialogue.

The first sentence of these two seems reasonably easy, except for the

²⁰<http://www.dvd-basen.dk/>

²¹http://www.dvdangle.com/reviews/clerks_cs.html

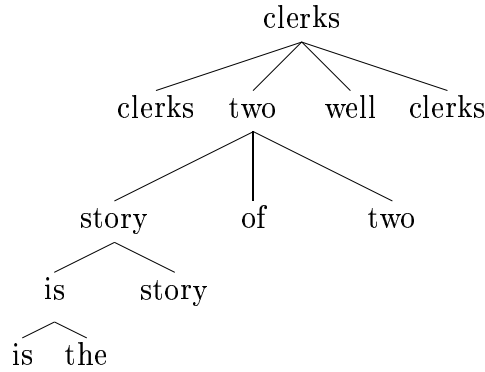


Figure 8.16: Parse for “*Clerks is the story of two, well, clerks*”

interjection “*well*”. This sentence receives a totally nonsensical parse however and it is clear that our GIM is not able to handle such an exceptional structure (Figure 8.16). The 2nd sentence seems more difficult, but the parse it is attributed is in fact far from bad (Figure 8.17). The parse correctly creates a separate constituent for the subject, even marking the apposition. And even though the VP features some unnecessary embedding, the basic constituent boundaries seem to have been respected.

Next we look at two sentences from the technical description of the DVD, which should feature some typical constructs and terminology:

1. The picture is quite grainy and there is some dirt present.
2. The dialogue is clear and distinct with little hiss throughout.

The first sentence is given a decent parse in which the two conjoined parts are prominently featured (Figure 8.18). Some interesting constituent boundaries are proposed and while this structure is quite respectable, it is unlikely that the human annotators that created the WSJ corpus would have provided a structure that is similar. This means that, even though the F-score provides a good quantitative measure of a grammar induction method, it should not be overestimated as a measure of the quality of the induced grammar itself.

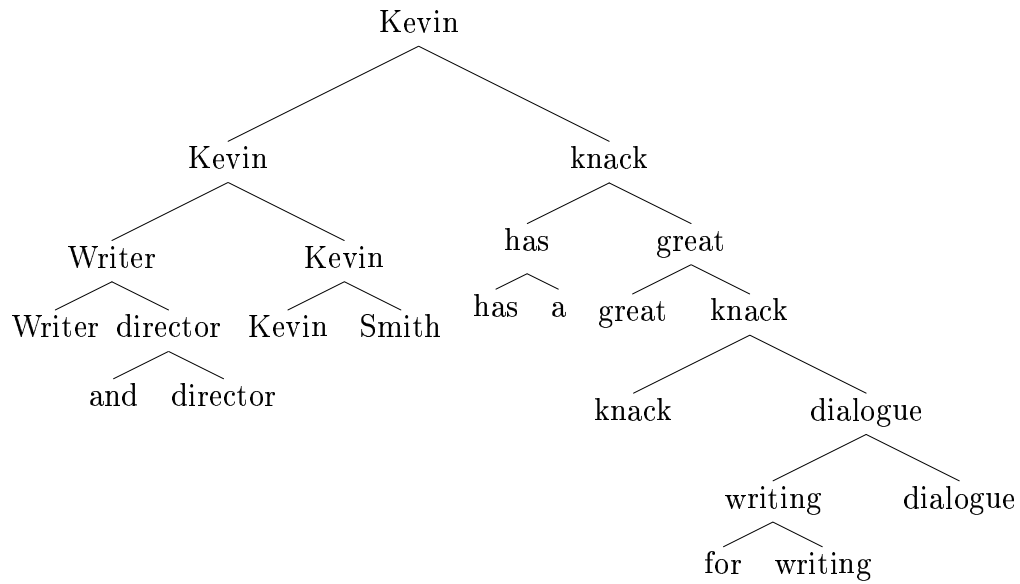


Figure 8.17: Parse for “*Writer and Director Kevin Smith has a great knack for writing dialogue.*”

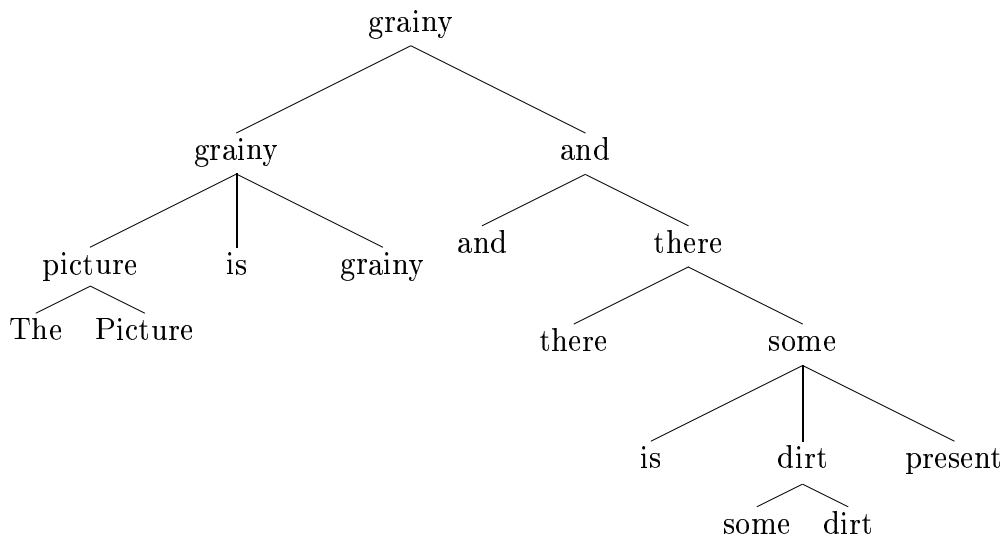


Figure 8.18: Parse for “*The picture is quite grainy and there is some dirt present.*”

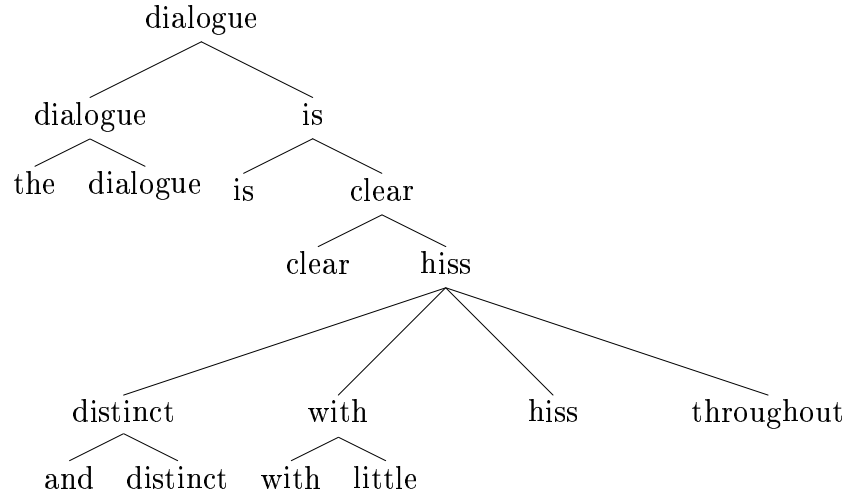


Figure 8.19: Parse for “*The dialogue is clear and distinct with little hiss throughout.*”

Although the 2nd sentence features common terminology such as *dialogue* and *hiss*, the attachment of the PP-structure seems difficult to retrieve, which is acknowledged in the parse in Figure 8.19: the subject is clearly delineated from its predicate, but the attachment of the PP-structure is indeed incorrect. Despite the high number of observations for this type of sentence, the GIM coupled with GRAEL-3 is still not able to find a good parse for this sentence.

Although there are many sentences that do not obtain a good parse using this method, there are also quite a lot of sentences that receive at least a partially good parse. We noticed that GRAEL-3 is in general able to obtain a good segmentation accuracy for NPs, but it does still seem very vulnerable to sparse data problems (cf. “*wel*” in Figure 8.17).

8.4 Advances and Concluding Remarks

In this chapter, we introduced the GRAEL environment as an unsupervised grammar induction method. We described a minimalist method to bootstrap syntactic structure throughout the society. Especially given its extremely simplistic approach, the results compared reasonably well to other

approaches, such as [Clark 2001] and [van Zaanen and Adriaans 2001]. Initially outperformed by these more elaborate systems, the GRAEL environment was able to narrow the gap.

There are still some research issues left to explore: Table 8.1 provided a wide range of possible instantiations, which uses different parsing methods at different moments in the GRAEL environment. Perhaps we are overlooking some obvious combinations which might increase GRAEL's optimization capabilities. Graphs like Figure 8.13 and Figure 8.14 also hint that performance might still be increased if we allow the society to run even longer. We will also need to extend the bigram-nature of our GIM so that it can consider a wider range of syntactic relationships. And there is also no reason not to use other unsupervised grammar induction methods such as ABL [van Zaanen and Adriaans 2001] or CDC [Clark 2001] to bootstrap a GRAEL-3 society.

We have also discussed that on the one hand, we wish to perform completely unsupervised grammar induction on words, but also the need to have some intermediate level of generalization. Perhaps a pre-processing phase that tags the words of a corpus with an (un)supervised method, might advance our grammar induction task. On the other hand, the structures provided by the grammar induction method, might also help to generate part-of-speech tags by itself or provide clues for the disambiguation task of tagging.

What the experiments with GRAEL-3 wished to prove is that it is possible to induce a ps-grammar in an unsupervised manner, using a very simple but effective approach. Current limitations do not seem inherent to phrase structure as a method of representing syntactic structure, as [de Marcken 1995] and [Yuret 1998] claim. Most unsupervised grammar induction methods investigated so far may have concentrated too hard on inducing structure by strictly adhering to maximum likelihood estimation, instead of looking at the possibilities of employing very basic information about the affinity between words, in the same way [Yuret 1998] used to induce dependency structures. By introducing a factor of randomness in the induction of grammars (cf. GRAEL-3B), we may also have found a way out of the local maxima that [de Marcken 1995] described.

The limitations of using GRAEL-3 as a grammar induction method can

therefore be situated in the limited performance of the grammar induction method itself, rather than the GRAEL system as an optimization technique. The research described in this chapter has given further evidence for GRAEL's ability to optimize grammars. Even if they are vastly deficient, the distributed approach to grammar optimization always seems able to squeeze some extra percentiles out of the grammars, simply by having them interact with each other.



Part III

The Emergence of Grammar

The Lord said, "If as one people speaking the same language they have begun to do this, then nothing they plan to do will be impossible for them.

Genesis 11: 1-9

9

Modeling the emergence of Compositional Language

In this chapter, we will provide an overview of some research efforts that have tried to model the emergence of syntax in a computational context. We will mainly focus on what kind of information is provided to the system, i.e. what is presupposed, and see what kind of grammatical structure emerges from that. In Chapter 10 we will then present our implementation based on the previously defined GRAEL environment and key elements from the research described in this chapter. We will discuss the individual approaches in Section 9.1¹ and compare them to each other in Section 9.2.

¹A great deal of gratitude is due to [van Trijp 2003] which laid the groundwork for this overview.

9.1 Computational Simulations of the emergence of syntax: the systems

There are three major players in the field of research that tries to model the emergence of compositional language in a computational context. This section will discuss these respective systems and finish off by briefly looking at some other related research efforts.

9.1.1 Negotiated Communication

John Batali has published two major publications that describe two rather different approaches towards the computational modeling of the origins of compositional grammar. [Batali 1998a] describes a society of agents equipped with a recurrent neural network that can link strings of sentences to a limited set of meanings. [Batali 2002] on the other hand uses an exemplar-based approach to allow a society of agents to develop a mapping of sentences to complex meanings.

[Batali 1998a]

Batali adopts a non-nativist point-of-view by hypothesizing that language can emerge in a population of agents that do not need any innate linguistic capacities, but some general cognitive abilities that allow them to build a structured mental representation of a situation. The capacity to do so is usually attributed to animals with a superior intellect and has in fact been linked to the development of language in early hominids. Even though Batali does not claim to provide a realistic simulation of the actual origins of language, he aims to prove that language can emerge without presupposing an innate language acquisition device.

To study the emergence of grammar, [Batali 1998a] first defines a meaning space, featuring the different possible meanings the agents can express. This meaning is presented as a sequence of 10 binary digits, of which the first six constitute the predicate and the remaining four the referent. 100 different meanings can then be represented as follows:

Predicates		Referents					Example Meanings	
<i>values</i>		sp	hr	ot	pl		<i>values</i>	
011001	happy	1	0	0	0	me	0110011000	<i>me happy</i>
100110	hungry	1	1	1	1	all	1001101111	<i>all hungry</i>
...	011001	<i>all happy</i>
						

A society of agents is then initialized, in which each agent holds a meaning vector (a string of 10 binary numbers between 0 and 1) and a recurrent neural network that can link strings from the signal space (random elements from {a,b,c,d}) to one of the 100 meanings from the meaning space. Each round, one agent is selected as the *learner* and 10 consecutive other agents are selected as the *teachers*. Each teacher will convey a string, expressing some meaning to the learner. This string is run through the learner's recurrent neural network: if the string causes the neural network's output layer to trigger the meaning the teacher intended, the communication is successful. Learner and teacher are considered to hold the same meaning if there is less than a 0.5 difference between their 10-number meaning vectors.

If the communication is not successful, i.e. the meaning vector suggested by the learner is different from the one the teacher intended, back-propagation in the learner's neural network causes the weights to be altered to incorporate the negative evidence for the meaning-signal relationship proposed by the learner.

When a teacher wants to produce a signal to a learner, he would in effect need to reverse the direction of the neural network. This is however not trivial. Actual production is done by having the agent produce a string that, if run through his own neural network, would trigger the correct meaning vector.

The experiments show that after 15.000 rounds, the agents use the same string sequence for most of the meanings in the meaning space. The agents seemed to have converged on a grammatical system. Batali points out that this kind of convergence does not necessarily indicate the actual emergence of grammar, as the agents might have settled on a set of strings that is not compositional in nature. But Batali is able to show that there are systematic tendencies: the strings produced by the agents can be interpreted as having a stem, consisting of two to three letters which expresses the attribute and a

modifier expressing the referent. This referent is often the same regardless of the predicate, indicating that the agents have converged on a common way to modify it.

Another experiment was conducted to see if agents would be able to create novel meaning combinations, by holding out 10 meanings. One agent was selected to produce sequences for the novel meanings, after which another agent was asked to interpret them. The experiment showed that the agents used the regularities, learned in the society, to convey and interpret these novel meanings, which had previously been unavailable to them, with reasonably good accuracy.

[Batali 1998a] shows that agents can converge on a compositional system to express a limited set of meanings. But it is indeed this restricted meaning space that is problematic for the appreciation of this experiment: with only one distinction to make (predicate vs referent) and a small pre-structured meaning space, it might be the case that the neural network approach provides an obvious lead towards convergence. Providing the agents with the explicit cognitive capacity to link a small signal space to a small meaning space, might provide the society with an unrealistically big advantage towards convergence. But even though the grammatical system that originated in the society seems limited to a collection of inflection rules on the predicate, the experiment does provide a basic method for the emergence of compositional language without reference to explicit innate linguistic capacities.

[Batali 2002]

Batali addresses most of the problematic issues of his 1998 paper in a new set of experiments, featuring a huge meaning space and a new approach to modeling mental processing: neural networks are replaced by an exemplar-based approach, akin to Similarity-Based approaches (cf. MBL in Chapter 2). The hypothesis remains unaltered: Batali suggests language originated as a means to externalize and communicate past and planned situations stored in memory using general cognitive mechanisms.

Rather than using neural networks, or a rule-based approach, Batali adopts *exemplars* as the basic building blocks of mental processing. For language, these exemplars can be used directly to convey some kind of meaning, or be modified to construct new analyses of the mapping between signal and meaning. Learning, according to Batali, involves the resolution of the com-

petition between a large set of exemplars, of which only a consistent set of frequently used items are retained over time.

In this experiment, Batali expresses the meaning of a situation in a *formula set*: a formula consists of a *predicate* and one or two numerical variables: e.g. (goose 1) or (tickled 1 2). The variables in the formula set designate the participants. Batali defines two types of formulas: *properties* (e.g. (goose 1)), that only take a single argument and describe a name for an animal or an intransitive verb, and *relations* which define two participants to some kind of course of action. With 22 properties and 10 relations, the meaning space in this model comprises of 2.3×10^{13} formula sets for the agents to choose from.

The following formula sets describe a meaning representation for “*The goose sang*” and “*The goose tickled a cow*” respectively:

- {(goose 1) (sang 1)}
- {(goose 1) (cow 2) (tickled 1 2)}

Batali also allows the agents to not only manipulate the sequence of words (*strings*) that they produce to express this meaning, but also the mental representations of a situation (*formula sets*) itself to represent more complex situations. This is done by taking the union of two formula sets. The following unified formula set expresses the meaning “*the singing goose tickled the cow*”:

- {(goose 1) (sang 1) (cow 2) (tickled 1 2)}

The mapping between the formula set expressing some kind of meaning and a string of words is performed by a data structure called a *phrase*. Such a phrase may simply store a 1-to-1 mapping between signal and meaning, but may also present an analysis of how a signal is mapped to a meaning and vice versa. Batali defines two types of phrases: *tokens* (Figure 9.1 left), which only contain a formula and a string, and *complex phrases*, which represent a structural analysis of the mapping from a string to a meaning on the basis of its constituents (Figure 9.1 right).

When producing or interpreting sentences, agents can use these stored phrases, but they can also compile new phrases from different phrases or

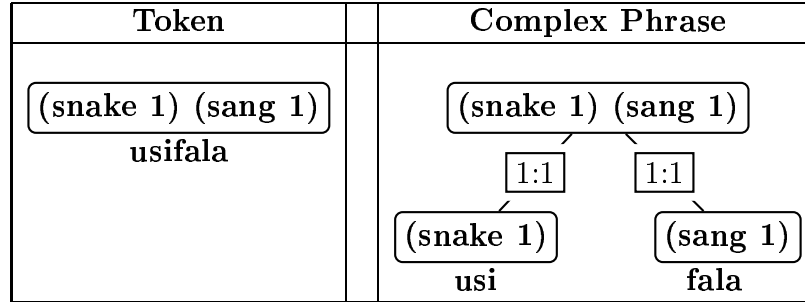


Figure 9.1: 2 Different Types of phrases

build their own phrases. They differ from each other in the *cost* value that is attributed to them, which expresses their preferability for production/interpretation. In communication, agents will also observe other agents map a meaning into a signal and will record these phrases in their own memory. Phrases observed in other agents are called *exemplars* and they will also be attributed a cost figure based on their value in subsequent communicative attempts.

The following example describes an idealized situation of an agent inducing compositional language out of communication with other agents. Initially, the agent is without phrases, but he observes another agent using the string *usifala* to express (snake 1) (sang 1). He will then consequently record this phrase/exemplar as a token (cf. Figure 9.1). Next, he might observe another agent use the string *usifalaozozj* to express (snake 1) (sang 1) (chased 1 2). The agent has no exemplar that can map the string directly, but he does have a phrase that maps the first seven characters. The agent can then induce two new exemplars: a token and a complex phrase (Figure 9.2). Consequent observations may cause the agent to create new exemplars by replacing sub-phrases of existing exemplars, as well as renaming the variables of exemplars or even create new tokens.

Obviously, this kind of processing will cause the agents to consider a wide range of different exemplars to choose from. The cost value attributed to them, however, introduces a way for the agents to select the most preferable phrase. An exemplar can be re-enforced by reducing its cost value, if it is used in the phrase an agent constructs to record a learning observation. Exemplars with a low cost value will typically be shared by many more agents in the

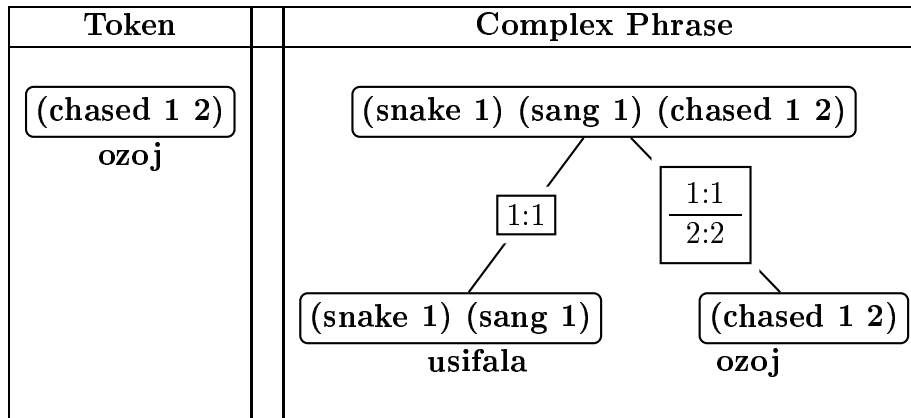


Figure 9.2: 2 Different Types of phrases

society, thereby increasing its observed frequency and thus lowering its cost value. The cost value of exemplars may also be increased if it is inconsistent with other observed exemplars. Finally, exemplars that have not been used for 200 runs are removed from the agent's mind.

The experiments show that the exemplar approach provides an interesting model for the emergence of compositional language. In one experiment, the society developed a kind of reflexive pronoun, others fixed word order and the marking of inversion. The result was different for most experiments, but some kind of grammatical system was apparent in all of them. Batali expresses the convergence of his system in a communicative accuracy figure²: almost all societies showed a high degree of convergence after a while. After the agents had learned to map the meaning space using a set of phrases, the society converged to a negotiated set of phrases that is shared by most agents.

[Batali 2002] improves over [Batali 1998a] by using a very large meaning space, which provides a much harder task for the agents. Impressive though the results are, there are still some problematic issues. In [Batali 1998a], Batali used a neural network, which provided an implicit bias for the agents to map the signal to the meaning it tried to express. The symbolic approach in [Batali 2002] replaces this bias with another approach that may presume

²The communicative accuracy figure is also used to compute the cost of a phrase.

too much linguistic capacities in the agents. Even though Batali argues that his approach only presupposes cognitive mechanisms of general utility that allow them to memorize structured representations of certain situations, the agents seem to be able to do more than just that. Whereas the substantial manipulations the agents are allowed to make on the formula sets, can be considered as being driven by the aforementioned general cognitive mechanisms, it is a bit more problematic to assume the same for phrases, which map a signal to a meaning and vice versa.

One can indeed imagine that primitive hominids are able to mentally represent the meaning space in a structured fashion using primitive cognitive control structures, but the intricate use of phrases that is apparent in Batali's system is more problematic from this point-of-view. And it is exactly these phrases that are the key to the emergence of grammar in the society. To bootstrap syntactic structure in the agents' minds, Batali needs to define a large number of manipulations and extrapolations that the agents are allowed to perform on phrases. But this presupposes an intermediate mental level between meaning and language. This would be justifiable if it were limited to a mere engineering trick that translates meaning into syntactic structures using very simple principles. But the way this intermediate level is implemented in [Batali 2002], makes it paramount to the emergence of grammar itself, controlling the way in which structured phrases are built. As it is, the use of phrases crosses the line from being a general cognitive mechanism to a more or less implicit innate linguistic capacity.

Also, it is not clear how the exemplar-based approach in [Batali 2002] incorporates a very essential aspect of natural language: irregularity. The use of phrases allows the society to converge unto a common grammatical system that is shared among agents, but it appears that this grammatical system is too clean cut to constitute an approximation of grammar in natural language.

The main merit of [Batali 2002] lies in his implementation of a symbolic, and therefore well interpretable method for modeling the emergence of grammar in a computational context. It should also be noted that the agents do not share structured knowledge explicitly, as is to some extent the case in the experiments described in Section 9.1.2. In Batali's system, the agents are able to converge to a grammatical system simply on the basis of linear strings of words and the (unstructured) meaning they represent. It is also

noteworthy that, despite a very strong bias towards compositionality in the agents' mind, the agents converge to a different kind of grammar in each experiment. Batali's experiments therefore prove his hypothesis that grammar can emerge in a society of agents that only require cognitive capacities of general use, even though the latter point is open to debate.

9.1.2 Iterated Learning

In this section, we overview the basic methods followed by Simon Kirby, Henry Bright and James Hurford, as they present the emergence of grammar from roughly the same point of view. We will first look at the iterated learning model developed by Kirby [Kirby 1999; Kirby 2000; Kirby 2001; Kirby and Hurford 2001; Kirby 2002a; Kirby 2002b], which he expanded on with Brighton [Brighton and Kirby 2001a; Brighton and Kirby 2001b; Brighton 2002]. James Hurford usually provides a more theoretical backbone for these experiments in publications such as [Hurford 2000].

[Kirby 98-01]

The iterated learning model, originally conceived by Kirby, tries to expand on the experiments of [Batali 1998a] by introducing a symbolic meaning space and a generation-based system. Developed over a number of publications, the iterated learning model has seen several adaptations to make it a more realistic computational model for the emergence of language, culminating in an entropy-based approach described in [Brighton and Kirby 2001b]. In this section, we will highlight the main features of the iterated learning model and see how it is used to model the emergence of compositionality.

Kirby states that the emergence of language is influenced by three complex adaptive systems: (1) *learning* in which children try to make sense of the observations around them during ontogeny, (2) *cultural evolution* which accounts for the fact that languages change over time and (3) *biological evolution* which provides humans with the cognitive capacities to process an intricate system such as language. Kirby hypothesizes that the emergence of compositionality can mainly be situated by the interaction of (1) and (2), provided (3) has given us the basic cognitive capacities to develop mental structural representations.

This matches Batali's views closely, but the iterated learning model that Kirby develops is quite different: it tries to explain the emergence of compositionality on the basis of the *transmission bottleneck*. If language is to be transmitted by cultural evolution from a parent to his offspring, the language should be easily learnable and therefore be compositional in nature, rather than holistic. The iterated learning method therefore provides a generation-based approach that ensures that language will develop over time in such a way as to make it easier to learn.

The iterated learning model holds two agents, one of which is a *teacher* (adult) and the other the *learner* (child). Initially, the teacher will present a randomly chosen meaning to produce signals for. The meaning space consists of two components: a number of actions and a number of objects/persons, while the signal takes the form of symbolic characters. For example: the teacher presents the signal "xkkq" to the hearer and its related meaning *loves(mary, john)*. The learner will store a rule that matches the signal to its meaning:

$$s / \text{loves}(\text{mary}, \text{john}) \rightarrow \text{xkkq}$$

As the teacher presents a whole set of utterances to the learner, a transmission bottleneck becomes apparent: the learner may indeed store the sentences and their meaning as a holistic language, but he will not be able to capture any other meanings than the ones originally provided by the teacher. The learner is therefore provided with an incremental grammar induction algorithm which generalizes over the rules: the learner may generalize over a pair of rules in the grammar and yield a generalized version of this rule, thereby introducing a notion of compositionality and therefore the ability to process new meaning/signal pairs.

After the learner has applied the induction algorithm on the data provided by the teacher, the latter dies and the former takes his place. He will then produce signals for a new set of meanings to a new learner, by finding the closest match for the meaning/signal pair he is trying to convey and fill in the blanks with randomly generated strings.

The experiments show that this kind of approach indeed allows the agents to develop a compositional language. The first generations will feature a largely holistic language, but sudden changes then allow syntax to emerge rapidly. On the basis of signals like *gjhftejm*, representing the meaning ad-

mires(mary, john) and *gjhftejwp* (loves(mary, john)), agents indeed develop rules such as:

s / p(x,y) → gj A/y f A/x B/p
 A/Mary (h)
 A/John (tek)
 B/loves (wp)
 B/admires (m)

Kirby also introduces a frequency bias into the system, to bootstrap the use of frequently used, but irregular exceptions and also allows the agents to produce imperfect sentences, introducing noise into the system. A principle of least effort is also implemented, which provides the agents with a preference for shorter strings. Despite these obstacles, compositional language is still able to emerge in the iterated learning model.

With these experiments, Kirby claims to have proved the fact that syntax can emerge, simply as a by-product of iterated learning: it is the transmission bottleneck that requires the language to be compositional in nature and therefore more easily learnable. But there are some problematic issues to the iterated learning approach as a computational model for the emergence of grammar.

A first minor issue is that the iterated learning method seems to involve *batch learning*: the grammar induction mechanism seems to process the data in one sweep (cf. *batch learning* vs *incremental learning* in Chapter 2). This does not constitute a realistic model of human language learning, but it does not seem impossible to transform the iterated learning method into an incremental model.

Whereas the agents in the model described in [Batali 2002] were presented with a signal and a linear meaning, the agents in the iterated learning model have the ability to mind-read and retrieve the entire structured mental representation. Clearly, this gives them an advantage in developing a compositional system that maps these meanings into signals. This is not necessarily problematic though, if there is no explicit connection between the structure of the transmitted meaning and the syntactic structures that emerge. Nevertheless, the explicit sharing of structural knowledge constitutes a rather unrealistic communication model.

But the main issue lies in the grammar induction method, which provides a very strong bias towards compositionality. Kirby's hypothesis that compositionality emerges to resolve the transmission bottleneck, misses the point: even though compositional languages are easier to learn than non-compositional language, its actual origins lies in a very strong grammar induction mechanism. It may be a by-product of complex adaptive systems such as cultural evolution and learning, which is what Kirby indeed proves, but its actual origins are the result of an implicit innate linguistic ability to generalize over grammar rules. The transmission bottleneck therefore helps explain *why* compositional language could have emerged, but not *how*.

[Brighton 01-02]

[Brighton and Kirby 2001a; Brighton and Kirby 2001b] address some of the problematic issues we identified in the iterated learning model. The strong grammar induction mechanism is replaced by one based on the *Minimum Description Length* principle (cf. Chapter 8). [Brighton 2002] describes the further development of this approach. The meaning space of Kirby's experiments is replaced with an external environment containing a number of objects (i.e. communicatively relevant situations) that are internally represented as points in the agents' meaning space. Brighton thereby introduces a distinction between object and meaning which allows a different mapping between meaning space and objects to be created for each experimental run. This feature also allows for synonymy and homonymy.

The overall architecture of the iterated learning model remains largely unaltered: a teacher transmits a meaning and a signal to a learner. The latter will memorize the observed data and then induce regularities using a minimum description length (henceforth mdl)-approach, which will allow him to generate new signals based on his previous observations. If the learner does not hold grammatical knowledge that allows him to create a signal for a meaning, he can invent one.

The key difference with the previously described iterated learning model, lies in the grammar induction approach. Rather than using an approach that is biased towards compositionality, Brighton induces grammar using information theory measures, not unlike those described in [Clark 2001]. Given a set of observations, the agent will create a Finite State Unification Transducer (FSUT) which maps symbols to meaning. This will initially express

a holistic language, with a path for each signal-meaning pair. A generalization process can merge states and edges in the transducer that looks for common elements in the descriptions, introducing compositionality into the signal/meaning mapping.

Since many different mergings may be possible, there is a large hypothesis space of FSUTs. A beam search is implemented which considers the transducers in this hypothesis space: unless a transducer is not consistent with the observed data, it is evaluated in terms of the amount of encoding it requires (mdl). The transducer that yields the smallest encoding is then chosen as the grammar for the agent. The agent becomes the teacher in the next round and will produce new signals by searching for a signal in the transducer that is consistent with the meaning that the agent is trying to express. The degree of compositionality in an agent's grammar can be measured by looking at its *expressivity*: the more a transducer becomes compositional in nature, the fewer new inventions the agent will need to express some kind of meaning.

The experiments show that despite a very complex meaning space, a stable grammar does emerge. Brighton hypothesizes that the *poverty of the stimulus* problem puts pressure on the agents to develop languages that are easy to learn. This is in line with Kirby's notion of the transmission bottleneck, which requires the agents to develop compositional languages.

Even though Brighton's experiments involve a very complex meaning space, it does not entirely succeed in addressing the problems inherent to the original iterated learning method. There is still the unresolved issue of the structured meaning space. Brighton invokes the poverty of the stimulus-argument to explain how and why compositional language emerges. But providing the learner with a structured meaning does not exactly establish a poor stimulus.

Brighton does succeed in finding an alternative to Kirby's grammar induction method, which was geared to inducing grammatical concepts. But even though the method itself is different, the actual outcome is not. The explicit linguistic notions are replaced by a MDL method, which however provides an almost equally strong bias as a grammar induction method. Chapter 8 already discussed several unsupervised grammar induction methods that use the minimum description length principle to induce grammar. Not considering the problematic issues described by [de Marcken 1995], Brighton's

grammar induction method introduces a powerful linguistic bias that, even though hidden behind an information theoretic method, is just as problematic as Kirby's.

Brighton acknowledges the fact that his agents can be considered to have an innate LAD, that allows them to develop structure, but argues that “[t]he possibility of a design does not imply its occurrence”. Emergence of grammar is the result of the transmission bottleneck, which requires languages to be easy to learn and therefore compositional in nature. But the same conclusion we drew with respect to Kirby's experiments applies to Brighton's: compositionality may have indeed developed over time to make languages more learnable, but Kirby, as well as Brighton are only able to prove that this is so, if and only if some innate linguistic ability is present that is able to generalize over grammar rules, which begs the question: how did this ability develop out of the cognitive mechanisms of general utility in early hominids?

9.1.3 Language Games

The research conducted at the VUB AI-lab, models the emergence of language by interpreting a community of language users as a complex adaptive system, trying to develop a shared communication system. The research is subdivided to yield explanations for different aspects of language, including phonology, lexicon and meaning creation and grammar. Work on modeling the emergence of grammar is still in progress, but some interesting experiments have been presented nevertheless [Steels 1998b; Steels 1998a; Steels 2000].

The computational model presented in [Steels 1998b] differs from the previously described systems in that grammar is not seen as a formal set of rules that govern the structural aspects of utterances, making transmission between speaker and hearer an easier task, but as a way to map semantic meanings unto sentences. Steels thus adopts a functional [Dik 1997] and cognitive [Langacker 1987] point of view by considering grammar as a way to combine basic lexical items to express more complex meanings, rather than a way to minimize the entropy of a sentence. To allow for such a view, the agents are not only required to organize their grammars, but also the meaning space itself: initially the basic lexical items suffice, but as the need

for more complex meanings arises, grammatical principles need to be devised to allow for the expression of these meanings.

[Steels 1998b] describes such an experiment: the agents are provided with a general cognitive framework that allows them to record situations, recognize previously recorded situations and re-enact situations. A situation is encoded as a *schema*, that holds slots, restrictions on the fillers for those slots and a number of constraints on the overall schema. The experiment starts off as a discrimination game³ between two agents: a speaker and a hearer situated in a context. An object is selected and both speaker and hearer will record the distinctive features of that object as the meaning to be expressed.

If the speaker has no word to express this meaning, nor any means to recombine previously recorded words to express it, he can invent a new one. Whether newly invented, or retrieved from memory, the hearer will observe a word. If the hearer does not know this word, he will associate it with each possible distinct feature set he observes, allowing for ambiguity to remain (temporarily) unresolved in the lexicon. If some word presented by the speaker is incompatible with the meaning it triggers in the hearer's lexicon, both agents will remember the lack of communicative success it yields.

As the lexicon expands and the agents start to agree on a lexicon, the need to express more complex meaning may arise. Rather than inventing new words for each complex meaning, the lexicon will start to propose multiple word utterances. Grammar arises when the cognitive memory system in the agents intervenes in the mapping of these multiple word utterances to their meaning by recording the form of word groups as syntactic schema's like in the following example:

Schema-541	
SLOTS	(syn-slot-51 syn-slot-50)
DESCRIPTION SET	([syn-slot-50 syn-cat-75] [syn-slot-51 syn-cat-76])
CONSTRAINTS	((PRECEDES (>>syn-slot-50)>> syn-slot-51))
CATEGORY	syn-cat-77
USE	10
SUCCESS	3

³An instance of a *language game*, the notion of which we borrowed and adapted to describe communicative attempts between agents in a GRAEL society.

The slot fillers (**syn-slot-***) represent syntactic functions such as subject, object, ..., while the **syn-cat-*** elements relate to syntactic categories, like part-of-speech tags. But not only the form of word groups is recorded, but also the complex meaning being expressed by them is recorded as a schema:

Schema-542	
SLOTS	(sem-slot-51 sem-slot-50)
DESCRIPTION SET	([sem-slot-50 sem-cat-75] [sem-slot-51 sem-cat-76])
CONSTRAINTS	((CONJUNCTION (>> sem-slot-50)(>> sem-slot-51)))
CATEGORY	sem-cat-77
USE	10
SUCCESS	3

The semantic slot fillers might correspond to such semantic functions like agent, patient, ..., while the **sem-cat-*** elements might express semantic properties of the slot-fillers, such as [-human] and the like. Also, the association between the semantic and the syntactic schema is recorded, to make the mapping between the two possible:

Schema-271	
FUNCTION	Schema-542
FORM	Schema-541
MAPPING	((syn-slot-51 sem-slot-51)(syn-slot-50 sem-slot-50))
USE	10
SUCCESS	3

Let us return to the language games. When the speaker needs to communicate a complex meaning to the hearer that requires a group of words, his cognitive memory will be invoked to find associations for this word group. The inference rules will look in the schema's related to this association and will extrapolate the syntactic and semantic restrictions. If all the slots in the semantic schema are filled and the restrictions imposed by the description set are not violated, the syntactic constraints are enforced on the group of words and communicated to the hearer.

The hearer will search the associations that are compatible with this word group and consequently extrapolate the semantic schema. The meaning provided by this semantic schema is consequently compared to the expected meaning. The game is a success when these two meanings match. When a language game fails, new schema's, associations and inference rules can

be created randomly to accommodate the current observation. This entails however that early word order is highly likely to turn into fixed word order later on.

[Steels 1998a] studies on a more general level how complex meanings can be expressed through a collection of forms (e.g. word order, intonation,...). The basic idea is the same: grammatical means only serve as a way to combine basic lexical items to express more complex meanings. As forms are re-used and combined, a hierarchy is created and categories, semantic as well as syntactic can be distinguished based on the slot fillers in the schema's. These categories can however also be used to increase the systematicity of the grammar itself.

The experiments described in [Steels 1998b; Steels 1998a; Steels 2000] present an interesting view of the emergence of grammar. By interpreting syntax as a method to convey complex meanings with basic items, the psycholinguistically relevant link between syntax and semantics is made more explicit. The experimental results presented by Steels however, do not indicate whether more complex utterances can be built using this approach. Currently, the grammar seems only able to render a minor structural impact on the language. Whereas the output of the systems described by Kirby and Batali could be analyzed in linguistic terms, this holds less true for the experiments described by Steels: the grammatical concepts can be found in the agents' minds, rather than in their productions.

Steels does succeed in showing that grammar can emerge in a society of agents without the need for an innate LAD, as there is less of a linguistic bias towards compositionality in Steels' agents. It is indeed the case that the agents are equipped, only with a general cognitive capacity to record, retrieve and re-enact situations. One might object to the use of the schema's, but these appear to be general enough to be considered as primitive cognitive constructs. This capacity can be used to record a complex semantic situation, a complex syntactic situation and an association between the two. It is not quite clear if these methods will suffice to create the kind of structural language in the way Batali and Kirby have, but current results go a long way.

9.1.4 Other Approaches and Related Research

In this section, we will briefly overview some related research efforts. A more elaborate description of these systems is usually not warranted, as they either describe a partial solution to the problem, or are more limited in their overall scope than the aforementioned models.

The conditions for Syntactic Communication

The group around Martin Nowak has written some influential papers on the computational modeling of the origins of syntax [Nowak et al. 2001; Plotkin and Nowak 2000; Komarova et al. 2001; Komarova and Nowak 2001b; Nowak and Jansen 2000; Komarova and Nowak 2001a]. The most important one of these [Nowak and Jansen 2000] describes a mathematical model that can explain why syntactic communication came about. The holistic language of animals is compared to human (compositional) language, by relating it to the number of events that need to be referred to. A mathematical model is proposed that relates the size of the lexicon to the number of events that agents need to express. A holistic language is at an advantage when there is a relatively small number of events that need to be expressed, but as the number of required signals exceeds a threshold value, a grammatical system, making infinite use of finite means, becomes preferable.

Even though the mathematical model described by Nowak goes a long way in explaining *why* syntactic communication emerged, it does not really describe *how* it happened. It merely outlines what conditions require a species with increases cognitive capacities to develop syntactic communication. The model described in [Nowak and Jansen 2000] is therefore to a large extent reconcilable with the research of Batali, Kirby, Brighton and Steels. Particularly the latter's view that syntax is a way to recombine lexical items into combinations expressing more complex semantic constructs, seems compatible with the conditions [Nowak and Jansen 2000] puts forward. Steels however takes the extra step to hypothesize how exactly grammar might have emerged.

Subsequent publications by Nowak [Nowak et al. 2001; Komarova et al. 2001] however take a different stance to Steels, Kirby and Brighton, by as-

suming a nativist point of view. A Language Acquisition Device is assumed and different experiments are devised in which a mathematical model is used to describe the properties of the universal grammar contained therein, as well as the conditions that need to be met to trigger one particular grammar on the basis of an innate universal grammar. Since Steels, Kirby and Batali succeed in modeling the emergence of compositionality with far fewer a priori assumptions on the agents' cognitive abilities, the universal grammar point is rendered moot by applying the principle of Occam's Razor. None of the experiments succeed in providing an insight into how early hominids might have evolved from beings with general cognitive mechanisms, to beings with an innate universal grammar.

[Plotkin and Nowak 2000] however provides an interesting account of how concepts of information theory can help explain the evolution of language on a more general level. The experiments with the aforementioned mathematical model show that the *fitness* of a language increases exponentially with word length. This is related to Shannon's theorem which states that *for a given noisy channel, there exists a sequence of codes with linearly increasing codeword length such that the probability of transmission error decreases exponentially*. The emergence of syntactic communication may therefore be seen as a way to decrease the transmission error in a noisy channel. This is an interesting point of view, as it also describes a condition for a species to move from holistic to compositional languages: since animals only need to express a rather limited set of events, they do not need more than a collection of unambiguous holistic signals. Human language users have the cognitive ability to communicate many events, but given the limited amount of sounds our vocal system can produce, using a holistic language to communicate these events, would produce a set of long sounds, often similar to one another. A noisy channel will then prevent successful communication from taking place, as similar sounds would become ambiguous and long sounds unintelligible. But again this publication describes the conditions that cause compositional language to emerge, rather than modeling how it happened.

Grammatical Evolution: Principles & Parameters

Ted Briscoe describes a series of experiments in [Briscoe 1997; Briscoe 1998; Briscoe 1999a; Briscoe 1999b] in which he tries to model the evolution of

grammar rather than the origins, but we include his work here to provide another nativist point-of-view. Briscoe works with a society of agents in an evolving context. All these agents have been provided with a **LAD** in which a **UG** is specified. The UG itself is implemented as a categorial grammar incorporated in an inheritance network describing the set of possible categories. Roughly speaking, the paths through the network describe the parameters of the UG. The agents are also equipped with a parser that can be used to analyze strings and consequently adjust the value of each parameter. The novelty in Briscoe's approach lies in the Bayesian mechanism that considers parameters not as binary dip-switches, but as a continuous scale with a particular threshold point to trigger the desired parameter setting.

Some experiments were performed in which a society of agents is initialized with varying parameter settings. After each round, the agents are attributed a fitness-value, based on their communicative success. As the fittest agents reproduce, a dominant language will appear over time. The Baldwin effect⁴ will make sure that over time agents will be consistently born with compatible parameter settings to those of the dominant language.

Briscoe's experiments describe the application of the interesting hypothesis that parameters in a LAD should be considered as a continuous scale during acquisition and how language and the LAD can be considered as by-products of co-evolution in a society of agents. Briscoe does not however explain how (compositional) language might have originated in early hominids, nor does he intend to. Even though he assumes that the original LAD would only presuppose minor changes to general cognitive abilities, the previously described experiments hypothesize that no such adjustment is necessary to trigger compositional language. Firmly rooted in a nativist point-of-view of language acquisition, the appreciation of Briscoe's experiments tends to be dependent on one's conviction on this matter.

⁴[Baldwin 1896] describes how advantageous learned behavior may turn into innate capacities over time through Darwinist evolution, e.g. sheepdogs who seem to have the innate ability to gather sheep.

Grammatical Evolution: Competing Grammars

[Yang 2000] describes an alternative approach to Briscoe's P&P experiments. Rather than explicitly implementing a parameter-based approach, Yang considers a hypothesis space of competing grammars in our minds. Language acquisition consequently involves attributing weights to these grammars based on the linguistic observations. Rather than considering one possible grammar, a speaker can be considered to hold multiple grammars in his mind, all competing to produce and analyze sentences.

This point-of-view provides a very dynamic perspective on the evolution of grammars. It is able to explain grammatical changes, without presupposing actual mental changes and it can account for several natural language effects, such as dialects, sociolects and idiolects. But again [Yang 2000] does not give any insight into the origins of compositional language itself. It still presupposes innate grammatical knowledge, which is justifiable if we consider a general nativist point-of-view to language acquisition, but does not provide a realistic model of how language emerged in early hominids. Again, this was never the intention of the experiments to begin with. We have included this model here because it is able to explain the evolution of grammatical principles, using a very general notion of innate knowledge that is able to model linguistic properties that the usual P&P approach cannot.

Emergence of Tree-Adjoining Grammar

A more formal approach to modeling the emergence of grammar is described in [Allexandre and Popescu-Belis 1998a; Allexandre and Popescu-Belis 1998b]. In these experiments agents send and receive signals about the meaning space, with both signals and meanings being structured by a Tree-Adjoining grammar. This presupposes a cognitive capacity in the agents to apply structure to both. The meanings that are expressed are situations in a block-world domain in which objects (cubes, spheres, ...) can have characteristics (color, size, ...) and can be positioned in a certain way (in front of, on top of, ...) as well as compared to one another (size, ...).

A society of agents is initialized in which the agents have no names for any of the objects in the meaning space. In each round (dialog), two agents

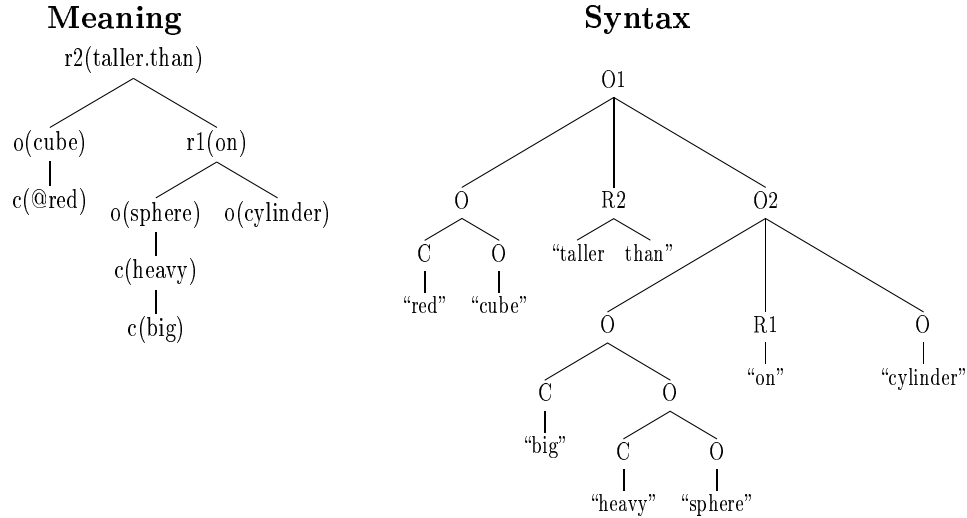


Figure 9.3: The Emergence of Tree-Adjoining Grammar

are selected and presented with a common situation. The sender uses his preferred lexicon and word order⁵ to send a signal which the hearer will consequently try to decode using his lexical and grammatical knowledge. If the dialog is a success, the parameters that were used, have their fitness increased.

Initially, the agents are presented with only the objects. A 5 agent society needs about 500 dialogs to establish unique names for these objects. When characteristics are introduced, the society needs 1200 dialogs to establish a unique word order for them, after which the two types of relations are introduced, which each takes the society about 9000 dialogs to converge on a common grammatical system.

The experiments show that the agents are indeed able to derive syntactic structures from structured meanings, as is illustrated in Figure 9.3. But this should come as no surprise, as there is already structure present in the meanings presented to the agents. It also seems there is an uncanny resemblance between the structured meaning and the syntactic structure, which indicates that the meanings that are presented to the agents may bias them towards a particular type of structure. This experiment therefore provides only a basic illustration of how syntax has emerged out of necessity

⁵if these are not available, they are randomly generated.

to describe more complex situations, which is in line with [Nowak and Jansen 2000] and [Steels 1998b].

Inexplicit Transmission of Meaning

Even though [Smith 2001] does not explicitly deal with the emergence of syntax itself, it does provide a very interesting solution to the problematic issues apparent in the works of Batali, Kirby and particularly Allexandre. Smith criticizes these systems for considering communication as an interaction in which not only signals are transmitted, but also unambiguous meaning constructs. This not only constitutes an unrealistic model of communication, but also reduces the significance the results of some of these systems, as the syntactic structure that is introduced mirrors the structural aspects of the meaning space.

Smith proposes an approach similar to [Steels 1998b], in which agents identify objects in terms of their discriminative properties in a context. The hearer agent does not know which object exactly the speaker is talking about, and will need to discriminate it in the context based on its own perception of the context. Smith goes one step further than [Steels 1998b] still, by not providing the agents any feedback on the communicative success of the language game. The internal lexicon is only developed by considering the probabilistic properties of the lexical items observed either as a speaker or as a hearer.

Another interesting novelty Smith introduces is the way in which a speaker decides which word to use to express a particular meaning. There is a lot of synonymy in the system, which causes the same meaning to be expressed by several words. Rather than looking at the most frequently used word to express a meaning, the speaker will find the word that he most likely would associate with that particular meaning, therefore concentrating on how the hearer will perceive the signal, rather than finding the most likely production.

Experiments show that even after many language games, there is still a lot of synonymy in the agents' lexicon, even though the agents do converge on a shared lexicon. [Smith 2001] constitutes a very interesting account of how distributional properties observed in communications alone can account for convergence, even without explicit meaning sharing. The large amount

of synonymy is probably due to the lack of feedback among agents, which causes a lot of alternatives to co-exist.

It appears that Smith's communication model might be driving things too far, effectively establishing an unrealistically strict communicative model: in real-life communication, participants usually do not need to distinguish the topic from the context using distinctive features as it is obvious what is being talked about, especially if the referent is visible to the participants in the communication⁶. Early hominids can be for instance considered to point at things they want to talk to, rather than internalize the distinctive features first to trigger the best lexical item for it.

And the lack of feedback does not seem a very realistic view of communication: if the participants indeed notice that they are talking about different things, this would become apparent from their behavior, causing them to either initiate another communicative attempt to get things right, or at least make a mental note of how that particular signal-meaning mapping has yielded a failed interaction⁷. Nevertheless, [Smith 2001] does establish an interesting framework for a communicative model that presupposes minimal meaning transmission as well as a model that can consider language, not necessarily as a means to communicate about directly visible referents, but as a way to externalize mental recollections. In this view, the referent is not available to the participants in a conversation and only the utterances themselves can provide an ambiguous clue. We will argue in favor of this view in Chapter 10, as it incorporates the essence of language itself and allows syntax to develop as an expressive module in its own right.

A Formal Account

One of the first notable attempts to study the dynamics of grammar in an evolutionary context was presented in [Hashimoto and Ikegami 1996]. The experiments described in this publication, look at the evolution of grammars on a very formal level, i.e. by evaluating them in terms of their position in the Chomsky hierarchy. The experiments involve a society of agents that

⁶This only holds true if both participants are adults, but does not translate to a context of child language acquisition.

⁷This is however not trivial, as it would not be apparent what factor in the communication causes the failure of the communicative attempt.

hold a grammar, which they can use to communicate with one another.

In a series of language games, each agent transmits a sentence (top-down process in the grammar), while the other agents try to parse it (bottom-up process). The fitness of an agent is computed on three levels: (1) speaking, how many and what type of sentences the agent is able to produce, (2) recognizing, how many and how fast an agent is able to recognize a sentence and (3) how fast on general an agent's sentences are being interpreted by other agents.

Grammatical evolution is introduced by allowing the agents to mutate rules in their grammars, which allows them to produce more sentences, but which may also affect some of the other fitness measures. The experiments investigated two types of evolutionary dynamics: (1) **module-type** evolution occurs when a rule is introduced that allows other rules in the grammar to generate almost twice as many sentences. This causes the agents to be able to recognize many more sentences in a short period of time. (2) **loop forming** evolution on the other hand introduces recursion in the grammar, which allows the grammar to climb up in the Chomsky hierarchy (to context-free grammars). Hashimoto hypothesizes that the higher the grammar climbs up in this hierarchy, the better it will perform.

In the experiments however, the grammars do not seem to climb any higher up the hierarchy, due to the synergetic behavior of agents. Some agents will form a mini-society in which they develop their own grammars, effectively allowing them to optimize their fitness functions on a smaller scale. This does not only weed out unfit agents, but also has the tendency to delete agents holding higher hierarchy grammars, which are however different from those of the miniature society.

Even though Hashimoto seems to describe this effect as an undesirable trait, it actually may constitute a realistic situation. Arguably the weakest point in the method presented by [Hashimoto and Ikegami 1996], is its adherence to the *competence* view of grammar. Grammars are deemed to be good if they can produce many sentences, the only touchstone of which are the other agents' grammars. But the initialization of these grammars, as well as the mutation operations on the rules are largely random in nature. A sentence is deemed grammatical if it can be interpreted by enough randomly compiled grammars, regardless of what kind of meaning it actually expresses

(cf. rote learning). The fact that a substratum evolves that disturbs the evolution of the grammars is therefore desirable in that they refuse to evolve into a state that allows them to become ultimately expressive.

Although [Hashimoto and Ikegami 1996] does not describe the emergence of grammar by itself, its formal account of grammar evolution in a computational context contrasts and in some ways complements the systems of Kirby, Batali and Steels. By totally disregarding meaning as a crucial factor in the development of grammatical principles, [Hashimoto and Ikegami 1996] show that through simple processes of mutation, a grammatical system by itself can evolve into a more powerful state, provided it can rely on group dynamics for self-organization.

Evolutionary vs Group Dynamics

Another interesting point of view is provided by [Zuidema 2000], in which the emergence of grammar is studied by looking at the interaction between group dynamics and evolutionary dynamics. Zuidema states that (compositional) language emerges at the cross-section of a wide range of influential factors and builds on the experiments described in [Hashimoto and Ikegami 1996] to prove his point.

The first experiment describes the genetic transmission of grammars without employing some kind of cultural transmission. The experiment shows that social patterns, even without inter-agent interaction do indeed influence evolutionary dynamics. Subsequent experiments study the different kinds of grammars that emerge, depending on social and evolutionary factors, as well as the circumstances that are required for the agents to come to syntactic communication.

The research described in [Zuidema 2000] presents an interesting bridging between the dynamics of evolutionary computing and the formal aspects of linguistic analysis and therefore provides an alternative to [Hashimoto and Ikegami 1996]. The experiments are however firmly rooted in the work of [Hashimoto and Ikegami 1996], who study the emergence of grammar on a very formal level, so that the analysis provided by Zuidema may not extrapolate very easily to some of the more realistic computational models for the emergence of grammar.

A connectionist Approach

[Tonkes 2001] describes an alternative approach to Kirby's model, in which the strong grammar induction method has been replaced with the general cognitive mechanisms of neural networks. Tonkes adopts the non-nativist view that this is all that is needed for (compositional) language to emerge. Grammatical principles emerge, similarly to Kirby's experiments, as a result of a transmission bottleneck which requires languages to be easily learnable.

A first experiment shows that languages that match the innate learning biases of their users, will have more chance to survive over time. This language is generated by the *encoder* of the agents (i.e. the neural network as a production mechanism) and its success is measured by the *decoder* of the agents (i.e. the neural network as an interpreter of sentences).

Subsequent experiments are described that study generalization effects in language. Languages can be observed to evolve to facilitate generalization in the decoder simply by evolutionary means, but also on the basis of a specific set of examples. A final experiment expands Kirby's iterated learning model by introducing it into a larger population. Structure can still be observed to emerge, even though the process is significantly slowed down.

One of the most important factors in the emergence of grammar however, seemed to be the amount of training data available to the learners (in other words: the learner's exposure to language). The society in Tonkes' experiments needed a critical amount of data to trigger compositionality. Communicative success could also be increased by providing each learner with a fixed set of examples, which leads Tonkes to hypothesize that language users are also exposed to a perhaps not fixed, but still very similar set of examples while acquiring language.

The biggest difference between Kirby's iterated learning model on the one hand and the neural network approach of both Tonkes and Batali, is the lack of an explicit transmission bottleneck in the latter. Tonkes however argues that there is an implicit learning bottleneck in that neural networks have the tendency to generalize on similarity and that it is therefore much easier for them to learn a regular language than an irregular language. This is indeed an important point to make. We have already stipulated that the transmission bottleneck in Kirby's experiments describes why compositionality occurs, but

not how. Batali and Tonkes seem to have found a way to join and situate the *why* and the *how* of compositional language in the cognitive domain, rather than in the communicative domain.

9.2 Computational Simulations of the emergence of syntax: General Tendencies

We can roughly divide the aforementioned publications into two categories: the first category describes properties of the emergence of compositional language, while the second more broadly tries to cover the acquisition of compositional language and/or its evolution over time. We present an overview of the systems described in this chapter using this dichotomy and a general subclassification:

Origins		
Negotiated Communication	Neural networks	[Batali 1998a]
	Exemplar-Based	[Batali 2002]
Iterated Learning	General Induction	[Kirby 1999; Kirby 2000] [Kirby 2001] [Kirby and Hurford 2001] [Kirby 2002a; Kirby 2002b]
	Information Theory	[Brighton and Kirby 2001a] [Brighton and Kirby 2001b] [Brighton 2002]
	Neural Networks	[Tonkes 2001]
Co-Evolution	Language Games	[Steels 1998b; Steels 1998a] [Steels 2000] [Smith 2001]
Group Dynamics	Competence Grammar	[Hashimoto and Ikegami 1996]
	Evolutionary Dynamics	[Zuidema 2000]
Mathematical Model	Neo-Darwinist Evolution	[Nowak and Jansen 2000] [Nowak et al. 2001]
	Information Theory	[Plotkin and Nowak 2000]

Acquisition/Evolution		
LAD	Mathematical Model	[Komarova and Nowak 2001a] [Komarova and Nowak 2001b] [Komarova et al. 2001]
	Bayesian Parameter Re-Estimation	[Briscoe 1997] [Briscoe 1998] [Briscoe 1999a] [Briscoe 1999b]
	Competing Grammars	[Yang 2000]
	Co-evolution	Tree-Adjoining Grammars

In the next chapter, we will describe experiments with GRAEL-4, our own skeleton model for the emergence of grammar, which will draw several elements from the above sources to propose a system that presupposes a minimal amount of cognitive mechanisms in the agents, as well as a minimalist model of communication with as little knowledge sharing between the participant as realistically possible. Let us therefore quickly overview these particular properties in the systems described in this chapter. Table 9.1 outlines the most important properties of all the systems that study the

actual emergence of grammar. Not included are systems that concentrate on the conditions for the emergence of grammar [Nowak and Jansen 2000] or the acquisition/evolution of grammar (cf. *supra*). There are basically two variables that determine the degree to which the models are pre-defined: (1) the cognitive mechanisms that are presupposed in the agents and (2) the amount of information that is shared in communicative turns.

These aspects are important to keep in mind if we are to propose a possible model of how compositional language might have originated in early hominids. The cognitive mechanisms should be as general as possible: on one end of the scale, it is ludicrous to suppose the agents have a fully developed LAD, as this is the kind of language capacity we are trying to investigate. On the other end of the scale, it would be equally ludicrous to attribute no cognitive abilities to the agents at all, as this would a priori render them unable to produce language.

One of the most important limitations to many of the systems described in this chapter, including Batali's and Kirby's, is the explicit meaning sharing that is evident in inter-agent communication. Following the 'gavagai' argument in [Quine 1960], meaning should not be considered as something that is readily available to participants in a conversation for the disambiguation of a signal. And even if we would disregard the 'gavagai' argument and suggest that a single object can unambiguously be referred to by pointing at it or some similar technique, it is not clear how one can visibly distinguish and unambiguously point at a complex state-of-affairs, as is the case in Kirby's, as well as Batali's systems.

The communication model should focus on the transmission of signals, while the shared meaning aspect should be played down. Again, if we consider a virtual continuous scale, we would on one end find a system as described in [Allexandre and Popescu-Belis 1998a] in which a shared meaning structure is completely available to the communicating agents, while [Hashimoto and Ikegami 1996] finds itself at the other end of the scale with a view of grammar that is totally exempt from meaning.

Using these two scales, we outline a very crude 2-dimensional space in Figure 9.4 in which the models described in this chapter can be situated. The aforementioned desirable traits our computational model should have, tries to minimize the "value" of each variable and therefore the more desirable

Publication	“Cognitive Mechanism”	Signal Transmission	Meaning Transmission
[Batali 1998a]	Neural Network	{a,b,c,d}	Bit-String
[Batali 2002]	Exemplar-Based	{a-z}	linear meaning vector
[Kirby 2001]	General Induction Mechanism	{a-z}	Structured Meaning
[Brighton 2002]	Minimum Description Length	{a-z}	Structured Meaning
[Tonkes 2001]	Neural Network	Bit-sequence	point in unit-interval
[Steels 1998b]	Schema’s	lexical items	distinctive features
[Hashimoto and Ikegami 1996]	Grammar	sentences	-none-

Table 9.1: Overview of computational models for emergence of grammar

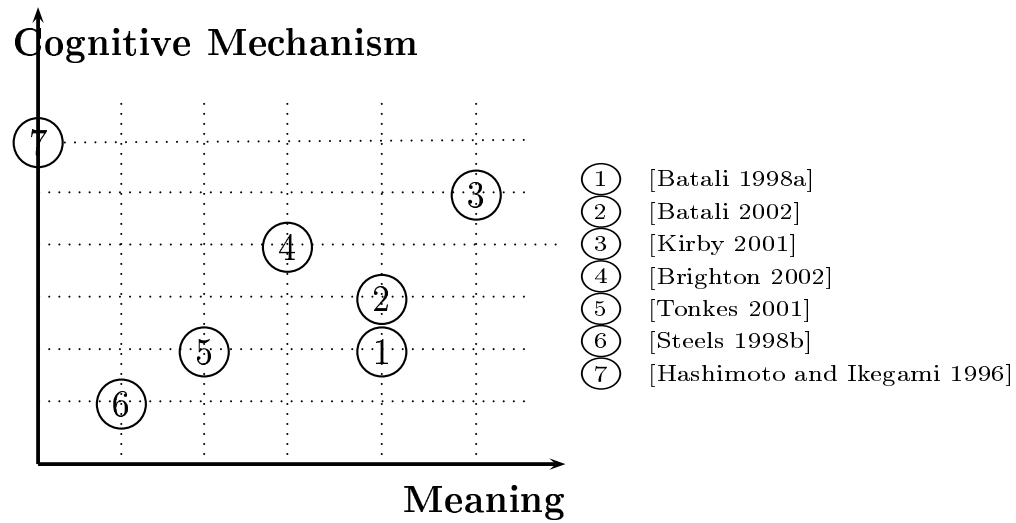


Figure 9.4: Assumption Scale for computational modes of the emergence of grammar

systems are located at the bottom left corner. This however does not mean that these systems automatically constitute good computational models for the emergence of syntax. We would need a third axis for that, but it is hard to define a “quality” measure for these models, as it is largely dependent on (a) what the author’s view on the actual role of syntax is and (b) the expressivity of the emerged grammar. As to the latter factor, it seems that the two systems that do find themselves at the bottom left corner in our graph, do indeed describe emerged grammars with limited expressiveness. On the other hand, as we move up towards the top of the axes, the grammars that have emerged seem to become more elaborate and reminiscent of natural language grammars. The following chapter will describe the basic outline of a model that breaks this apparent correlation.

This concludes our discussion of some of the most relevant pointers in the computational modeling of the emergence of grammar. In the next chapter, we will try to re-define the GRAEL-environment as such a method.

Even in the case of lifeless things that make sounds, such as the flute or harp, how will anyone know what tune is being played unless there is a distinction in the notes?

Corinthians 14:7-8

10

GRAEL-4 - Modeling the Emergence of Grammar

In this chapter, we pick the GRAEL-environment up where we left off: by further reducing the pre-defined elements in the unsupervised grammar induction method of GRAEL-3, we will propose a computational model of the emergence of grammar, similar to those discussed in Chapter 9. We will first discuss the features of this system in Section 10.1 and discuss the experiments in Section 10.2, after which we identify some problematic issues in Section 10.3. We conclude with some summarizing thoughts and pointers to extended research.

10.1 Architecture of GRAEL-4

In this section, we will propose a computational model that tries to limit the assumptions made on the agents' cognitive capacities, as well as on the content of their communicative transmission. Rather than pre-supposing a

direct link between semantics and syntax, cf. [Steels 1998b], we will concentrate on how rudimentary principles of syntax emerge from distributional aspects of communication, rather than the actual content that is being expressed. In this view, syntax is considered as a more or less autonomous expressive module in language production and understanding. Despite the fact that this view seems firmly rooted in Chomskian linguistic theory, we will take a deliberate non-nativist and performance-oriented stance in this matter.

10.1.1 The Innateness Discussion

We have already discussed some aspects of the innateness discussion in Chapter 9. There is a clear dichotomy between research efforts concentrating on modeling the origins of the language acquisition device that can account for the origins of compositional language [Briscoe 1998; Nowak et al. 2001] and research that tries to model the origins of compositional language proper on the basis of general cognitive mechanisms, rather than innate linguistic capacities [Steels 1998b; Batali 2002]. In Chapter 9, we have dismissed the former point of view on the basis of Occam's razor. Simply put: there is no need to assume specialized language capacities, if general cognitive ones suffice.

Many publications have already appeared on the innateness-debate and it is by no means our intention to add to the discussion. In this chapter, we will largely adopt the view presented in [Schoenemann 1999], as this provides an interesting falsification of the nativist stance from an evolutionary point-of-view. Schoenemann outlines a list of basic principles that hold true for any kind of evolutionary change, but are largely disregarded by the nativist field. These principles are used to evaluate a list of universal characteristics that are often attributed to a UG¹. Schoenemann suggests that the features of the UG that can be identified on the basis of cross-linguistic comparison, *“are so general in nature that they do not resemble rules, but instead are simple descriptions of our semantic conceptualization of reality. [...] there is no reason to suppose that syntax is anything other than conventionalized (i.e. invented) rules that allow languages to accurately communicate human*

¹Largely compiled from [Pinker and Bloom 1990].

semantics (the features of which may or may not have innate components)”.

[Schoenemann 1999] therefore sees syntax as the consequence of a need to communicate about an increasingly complex meaning space, which agrees with the nativist, as well as the non-nativist models described in Chapter 9. But whereas the nativist field assumes that this need somehow translated into an innate set of syntactic guidelines over time, the non-nativists are able to show that even without a genetic predisposition, syntactic principles emerge, but rather as a consequence of human semantic complexity, than constituting its actual cause.

10.1.2 The Naming Insight

A very important point with respect to the emergence of grammar is made by [Wray 1998; Wray 2000]. The classical view of early hominids' protolanguages is that of a language that can **name** objects, but is unable to express complex relationships between them (e.g. [Bickerton 1990]). [Wray 1998; Wray 2000] however suggests that protolanguage was holistic in nature and consisted of a large number of arbitrary, agrammatical signals expressing an entire state-of-affair.

This entails that utterances would need to be general enough, as very specific holistic messages would not be functional in everyday language. But this also means that these utterances would need to be specified by indicative gestures, such as eye-gaze or pointing, to disambiguate the general signal and as a consequence, the protolanguage could not be used in declarative statements to communicate about situations of which the referent was not immediately available. According to [Wray 1998] the holistic nature of the protolanguage explains a cultural and technological stagnation between 1.4 million and 100,000 years BC [Mithen 1996]:

“The holistic protolanguage [...] would stifle its own further evolution [...]: specific naming is unsustainable; without naming, declaratives have almost no purpose; without declaratives, information exchange is largely impeded; this minimizes technological and cultural innovation, rendering naming unimportant.”

The emergence of grammar, according to [Wray 2000], occurred simultaneously to the *naming insight*, so that there is no need to consider a gradual development of grammar:

“[...] naming is unleashed into a powerful cognitive forum that can immediately exploit referentiality by creating argument structure out of the juxtaposition of a word and a holistic utterance, and by segmenting holistic utterances to ‘identify’ new words and structures post hoc.”

[Wray 2000] further corroborates her claim by pointing at ‘living fossils’ of holistic phrases in present-day natural language, but also by referring to first-language acquisition. It is suggested that children also produce holistic utterances by imitation at the early stages of language acquisition. As they obtain the “naming insight”, they become able to segment these strings into smaller meaning units that can be juxtaposed in grammatical relationships. It is however not entirely clear from [Wray 2000] what kind of development triggers the naming insight and what method is used to segment holistic utterances into parts.

This view has been adopted by Batali, as well as Kirby in their experiments: early signals for meanings are completely holistic in nature. Grammar emerges as some generalization method finds regularities in the holistic strings, which leads to the *naming insight*, as well as grammar². This is also the reason why the agents in these systems need to present the meaning of the signal alongside the signal itself. It would otherwise be impossible to extract generalizations from the signals.

But even though this is a valid method if we are to model the emergence of compositional language in the way that is suggested by [Wray 1998; Wray 2000], we first need to question the degree to which we need to entertain this proposal. Considering language as a means to converse about visible referents, in the way that Kirby and Batali suggest, reduces language to a collection of **speech acts**, not as a general way to externalize mental state-of-affairs. Language indeed enables us to converse about past recollections

²This means that the two operations are considered to occur simultaneously, which contrasts the widely adopted view that grammar requires an a priori naming capacity.

in such a way that the referent does not need to be available at the time of speaking.

Furthermore, the meaning structures that both Batali and Kirby put forward are often so intricate, that simple methods such as eye-gazing or pointing would not be able to provide an unambiguous referent³. So if we adopt the view that compositional language evolved from a holistic protolanguage, we are not only diminishing the functionality of language from a pragmatic point-of-view, but we are also making a paradoxical assumption in that we do consider the ability of these hominids to mentally process structured items, but that this ability somehow does not extend to the linguistic domain until some kind of naming insight occurs.

[Wray 2000] answers this question by saying that the hominids do not feel the need to apply this kind of structure on their language, since their holistic utterances will do. But as the need for linguistic creativity becomes apparent, they somehow gain naming insight, which also allows them to segment the holistic utterances into re-usable parts. But this begs the question: how does this segmentation take place? Since [Wray 2000] states that the utterances of the holistic protolanguage are by definition not dividable into discernible parts, the segmentation and therefore the mapping between name and referent should be arbitrary.

But unless there are discernible parts in holistic utterances, there would be no reason to assume a correlation between the segmentation properties the naming insight incurs and the emergence of grammar. Linguistic generalization in for instance Kirby's iterated learning model, occurs when two rules can be generalized into one rule on the basis of the observed data. In the initial phase, this assumes that a holistic utterance is composed in such a way that there are in fact distinguishable components, which can be turned into names and consequently into grammatically re-usable items. To generalize over linguistic observations, there should indeed be enough evidence to turn part of a holistic utterance into a name. But this is supposed to be impossible, following the definition of holistic utterance as non-compositional tokens.

Furthermore, this does not make sense from an evolutionary point of view: if the original holistic structures are such that in a later stage of evolution

³This is already problematic for simple objects, as [Quine 1960] suggests.

they can be segmented to yield re-usable names, they should be quite lengthy. But [Wray 2000] agrees that the set of holistic utterances is necessarily limited and unspecific, so the hominids should suffice with rather short utterances. So why would the hominids invent lengthy holistic utterances, unless the length itself would indicate some compositional meaning, which necessarily presupposes an a priori naming insight?

And why would the emergence of grammar extract names from holistic utterances, if these names would need to be replenished with a new batch of arbitrarily chosen names anyway? It therefore seems reasonable to assume that the naming insight was available to hominids before they could consider re-usability mechanisms in grammatical constructs. But to assume that names were extracted from holistic utterances without considering a fully operational naming insight seems problematic, especially if we are to view language as a way to internalize the stimulus (word) - response (meaning) sequence⁴. We therefore adopt the classic view that agents have acquired the naming insight and therefore to a large extent compositionality in its own right. The GRAEL-4 experiments can therefore not be considered as a simulation of the emergence of compositional language, but as a simulation of how grammatical restrictions imposed on the juxtaposition of names, turns compositionality into a means of expression by itself.

10.1.3 GRAEL-4 Features

We will take a non-nativist point-of-view when considering a computational model for the emergence of language. The agents in the GRAEL-4 society will not be provided with explicit innate linguistic abilities, but with cognitive mechanisms that can be supposed to have been available to early hominids. [Batali 1998a] for instance suggested the agents' mind consists of a neural network that generalizes over the observed data. [Kirby 2001] did not simulate a "brain" in the agents, but did presuppose the ability to generalize over linguistic observations, while [Steels 1998b] provided his agents with the cognitive ability for planning and structuring.

Arguably the most controversial assumption relates to the previously dis-

⁴Comprehension: Stimulus=Word, Response=Meaning. Production: Stimulus=Meaning, Response=Word

cussed naming insight. Whereas Kirby and Batali start off with agents that do not have fixed names for objects, we will assume a basic lexicon in the agents' minds, containing concepts and names for those concepts. We do realize that this takes a totally different stance and this may also constitute an unrealistic model for the emergence of grammar to many researchers working in the field. But we have argued in the previous sections that we believe the existence of names to be a necessary prior condition to the re-usability of tokens and therefore grammar itself. The paradigmatic stance we take is that either one assumes that language starts out holistic so that stimulus (meaning/word) and response (word/meaning) always need to be in the vicinity, which renders the need for compositional language unnecessary, or one assumes that the agents can use basic, meaningful tokens to refer to things but not yet in a compositional manner such that compositionality itself carries meaning. Our stance is inspired by the acquisition of language in children, who typically are able to acquire an extensive lexicon, before they start producing multiple word utterances. As our outline of the GRAEL-4 system will show, we do however consider the co-existence of names and small holistic phrases, in accordance with [Wray 1998].

Cognitive Capacities

The agents in GRAEL-4 are assumed to have acquired the following abilities:

- **segmentation**: the agents are able to distinguish objects and their basic properties from the context and memorize them as separate entities
- **labeling**: the agents are able to provide words for these entities and memorize the mapping of meaning and word
- agents keep track of previous linguistic observations in their **memory**, as well as the frequency with which they've been observed to co-occur

The cognitive abilities are therefore limited to memorizing linguistic observations, their frequency and the meaning attached to the basic lexical items. Table 10.1 shows the lexical items the agents are able to express. We will discuss the importance of the subcategorisation property [\pm **animate**]

Objects				Attributes	
[+animate]		[-animate]			
agent0	agent1	house	fire	sad	happy
agent2	agent3	water	tree	hungry	stinky
agent4	agent5	stick	rock	Relationships	
agent6	agent7	seed	fruit	and	not
agent8	agent9	leaf	sand	runto	chase
dog	duck	flower	bed	throw	love
pig	bear	coat	egg	fight	hit
ape	horse	nut	lake	eat	kick
fish	snake	river	bone	know	fear
lion	tiger	society	food	see	offer
				catch	talk

Table 10.1: Agents' Lexical Items

in the next section. The right-hand side of Table 10.1 describes the attributes for and relationships between objects.

Similar to Kirby, we do not consider fitness functions. Even though language does provide a beneficial condition for the development of a species as a whole, we do not consider enhanced language abilities as a selective advantage for procreation, despite what [Rostand 1897] suggests. Our society will be generation-based, but this only means that agents will die and be born in the society: no knowledge will be passed on genetically from one generation to the next.

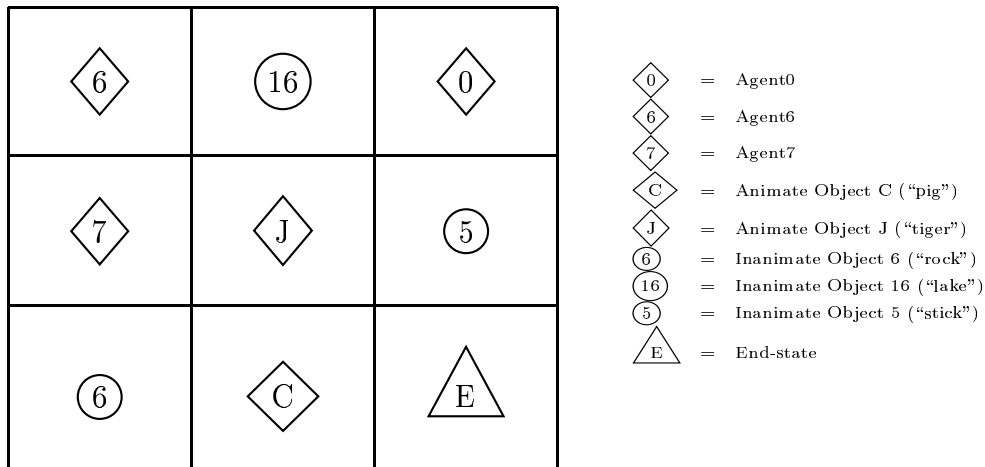
Contrary to the models of Kirby and Batali, the agents will not have the ability to mind-read, i.e. no explicit meaning will be presented to the agents: the hearer will need to distinguish the meaning of the utterance, purely on the basis of the words⁵ themselves and the word order.

⁵As names that express mental concepts, these can already be considered as meaningful units in themselves.

10.1.4 Communicating GRAEL-4 agents

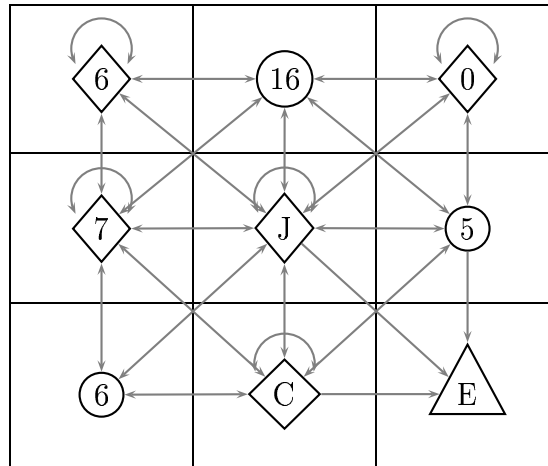
Let us now look at the underlying communication model for GRAEL-4. There are 10 agents in the society, who live in a closed world, surrounded by the objects outlined in the Table 10.1 (left). We assume our conversations to be a form of gossip, as suggested by [Dunbar 1996]⁶ and we therefore suggest that the reference for the utterances are not immediately available. In other words: what the agents convey are previously encountered, internalized situations, which are not visible to the participants of the conversation. We acknowledge that child language acquisition, and in all likelihood the emergence of language in early hominids, is rooted in the identification and naming of visible referents. But whereas previous methods trivialized the problematic aspect of the shared meaning in communication, we would like to oppose this in the GRAEL-4 experiment, to show that grammar can emerge in a society of agents without mind-reading skills.

Two agents are selected in the society to converse. The first agent recollects a situation in the form of a 2-dimensional matrix (between 2x2 and 4x4), consisting of cells with objects. The topic of the recollection is another agent (stated in the top-left corner):



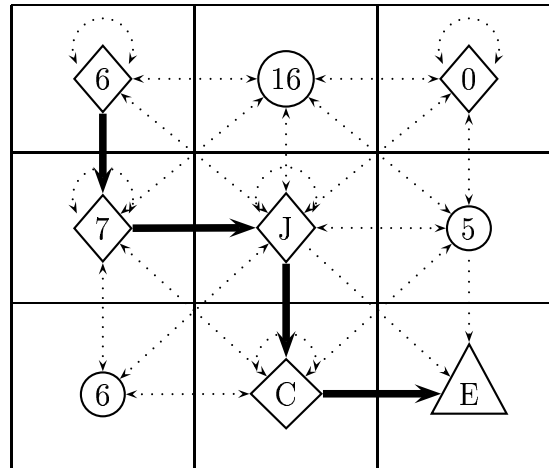
⁶Even though Dunbar may not agree with the time frame of the development of grammar we propose.

Diamonds represent animate objects, circles inanimate objects. We represent agents by diamonds that hold a digit, while animals have an alphabetic character. This matrix represents a state-of-affair in which three agents were present, as well as a *tiger*, a *pig* and there was also a *stick* and a *rock*, even though none of these may be particularly relevant to the situation itself. Next, we connect the cells with arcs, expressing specific relationships between the objects (cf. Table 10.1). Arcs leaving a [- **animate**] object are mere jump arcs, as we consider the inanimate objects unable to initiate a relationship. This restriction expresses the semantic distinction between agent and patient. Arcs inside a cell express attributes⁷, while pop-arcs leading to the end-state (triangle) do not carry a label and are uni-directional:



The matrix now expresses a scene in which the topic of the conversation, **agent6** in our example, was present. To prepare for the first utterance about this scene, the agent plots a path through the matrix, starting from the topic cell to the end-state cell:

⁷Trivially, these are not available for [- animate] objects.



This path currently expresses a seemingly serial sequence of events:

- `and(agent6,agent7)`
- `chase(agent7,tiger)`
- `eat(tiger,pig)`

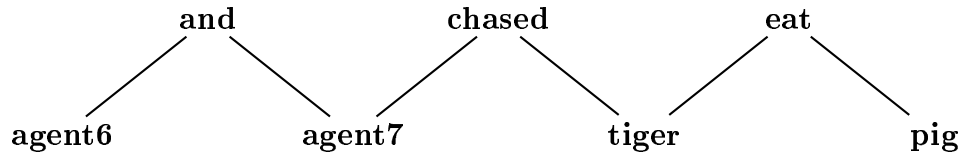
which we could interpret as meaning “*agent6 and agent7 chased a tiger that was eating a pig*”, even though this particular structure is not apparent as yet⁸.

An important point we would like to make is that the arcs in the above matrices, are not only able to describe relationships between objects, but also relationships between objects and the state-of-affairs surrounding the object. The `and(*,*)` property might for instance indicate that agent6 was simply in the neighborhood, while agent7 chased a tiger that was eating a pig. The serial sequence of events is therefore still ambiguous. Since there is no need

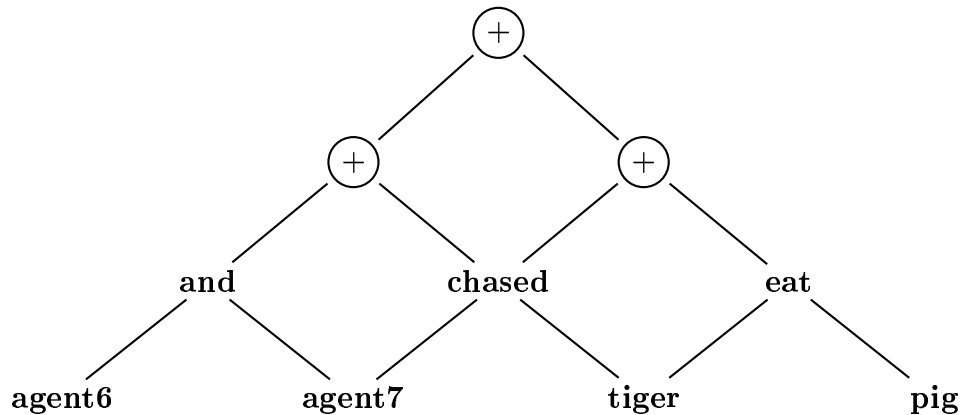
⁸Note that the words themselves do not guide the actual meaning of the sentence to be expressed. They are just used to facilitate interpretation. In other words: a tiger is just as likely to chase a fire as it is to chase a pig. We impose no meaning restrictions on the combination of words, as this could be considered to unfairly guide the emergence process.

to assume this kind of mental ambiguity, we next describe a way to apply an unambiguous structure to the events.

First we list the objects in the path in a serial manner, ordered by their occurrence in the path from topic to end-state and describe the basic relationships between them:

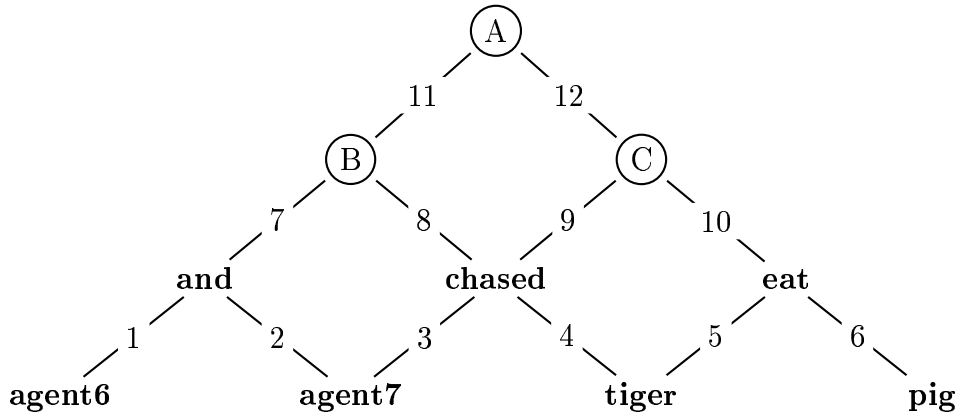


This structure can express the serial sequence of events, but not embedded meaning. We then superimpose a structure on these relationships which is able to express embedded meanings:

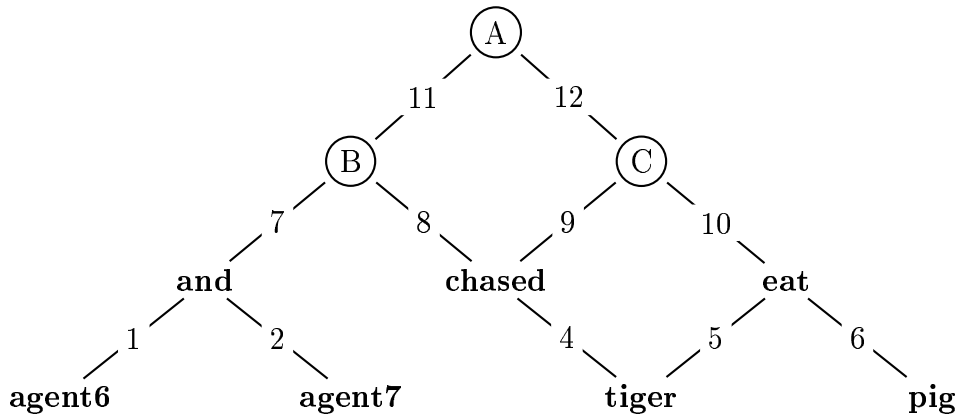


We then impose some formal constraints on this structure: (1) no item may be directly headed by two different nodes and (2) each node must head two leaf nodes, except for terminal nodes and empty nodes. The latter are just place-fillers and passers-through for superordinate attachment and can be deleted if they are not heading any nodes. Constraint (1) is checked in

a bottom-up fashion, constraint (2) in a top-down fashion. We have numbered the branches in the structure to illustrate how the application of the constraints might work:

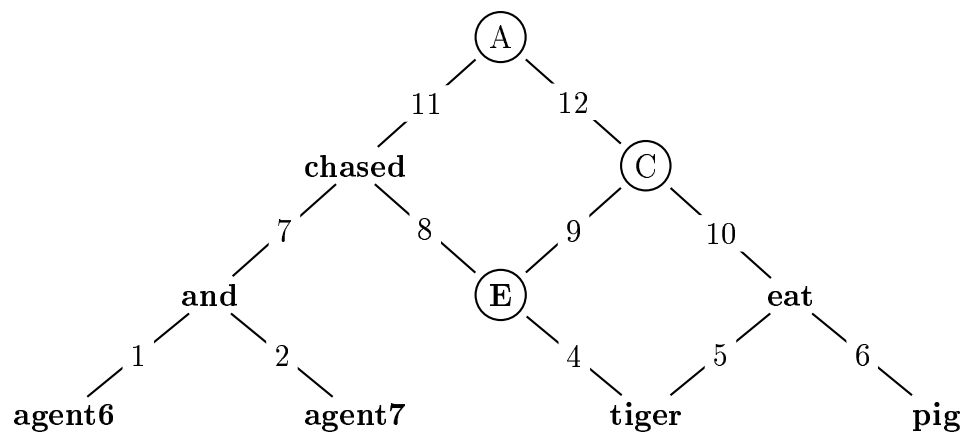


First we check constraint (1) and find two sets of conflicting branches on the lowest level: (2 3) (4 5). We randomly pick a set to resolve, in this case set (2 3), and delete one of the branches:

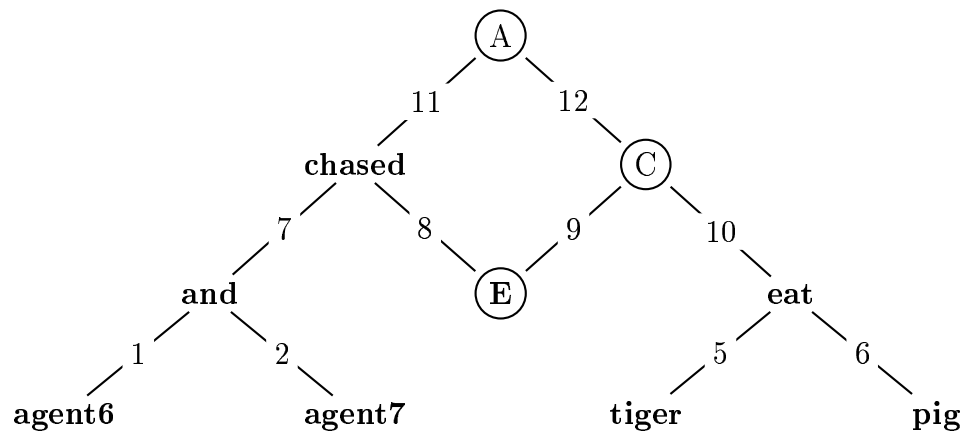


We then check constraint (2) and find that the node with label “**chased**” does not head two leaf nodes. Following a randomly chosen path, the label

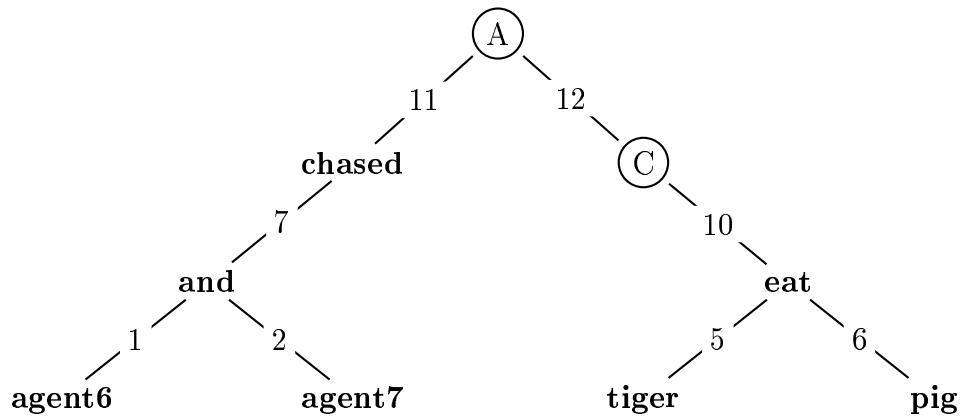
settles in an empty node that does meet the requirements set by constraint (2):



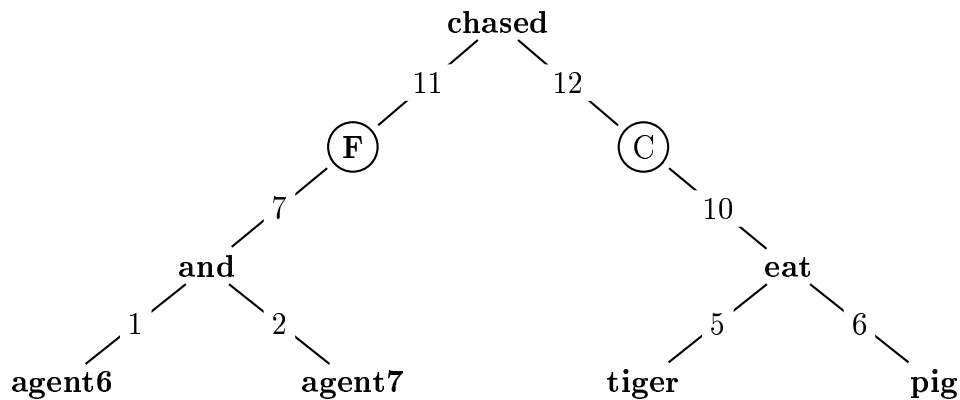
Constraint (1) is applied again and this time, we delete branch (4):



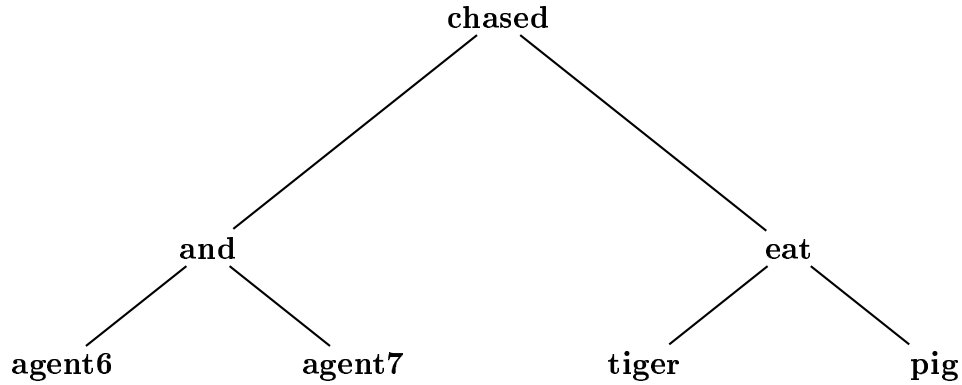
The empty node labeled **E** does not head any other nodes and is therefore deleted:



Constraint (2) is violated by the **chased**-node, which is consequently moved to a superordinate empty node:



We now arrive at a situation where neither constraint is violated: the message is considered to be fully structured now. The meaningless empty nodes now act as passers-through by collapsing the structure, yielding the following representation:



This structure represents the situation in which agent6 and agent7 together chased a tiger that had eaten a pig. But given the randomness of the entire approach, many different embeddings are possible. Using the same sequence of events, we are able to describe different situations and meanings, for example:

<pre> graph TD and[and] --- agent6[agent6] and --- chased[chased] chased --- agent7[agent7] chased --- eat[eat] eat --- tiger[tiger] eat --- pig[pig] </pre>	<pre> graph TD eat[eat] --- chased[chased] eat --- pig[pig] chased --- and[and] chased --- tiger[tiger] and --- agent6[agent6] and --- agent7[agent7] </pre>
<p>agent6 was there, as agent7 chased a tiger that ate a pig</p>	<p>agent6 and agent7 chasing a tiger, caused a pig to be eaten</p>

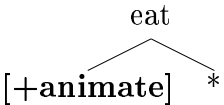
An important note to make at this point is that in no way are we considering syntactic processing, despite what the tree-structures and constraints might suggest. What we just described should therefore be considered as a mere engineering trick to apply (embedded) structure to a scene and should therefore not be considered as a realistic mental process. We assume that the agents have the ability to induce a structured mental representation of a scene, simply by ordering them according to salience, not by applying random

operations and constraints to mentalized structures. The process just outlined, is therefore only a way to construct an arbitrary structured situation, stored by the agents' cognition.

Linguistic Structure

Given the ability to recollect situations in a structured manner, we will now describe the way the agents are able to communicate about these. On the basis of the mental structure of the situation, the speaker agent will decide how he will create his utterance: he is in principle able to use a **name** for each node in the semantic structure (cf. Figure 10.1). He can in principle express the atomic objects and relations at the leaf nodes of the semantic structure (agent6,tiger,eat,...) using compositional phrases, but he can also use a holistic phrase, which expresses a node higher up in the semantic structure. In other words: he can express a complete situation with one name/token, but there are some restrictions.

[Wray 2000] suggests that holistic utterances in protolanguage were used to express complex semantic meanings. A token such as *tebima* can for instance be used to express a meaning “*give that to her*”. To simulate these remnants of a prior holistic language, we randomly define 50 holistic tokens at the onset of the society. These can be used to express general situations like *ate*([+animate],[−animate]) or *happy*([+animate]) or even *chased*(relation,*) and are defined in terms of the relations and attributes from Table 10.1. The names of these tokens are considered to be arbitrarily chosen, but for clarity of reference, we construct these names as follows:

Node	Name
 <pre> graph TD eat --> plus_animate["[+animate]"] eat --> asterisk["*"] </pre>	<i>eatanimateany</i>

The agents are considered to have picked up the meaning of these holistic tokens, as well as the mapping between the atomic mental concepts and their names during a lexical acquisition phase that is not simulated in these experiments.

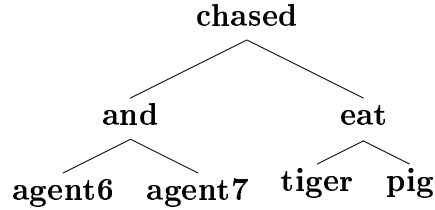


Figure 10.1: Structured Mental Representation of a Scene

So given a semantic structure like the one illustrated in Figure 10.1, an agent can express the leaf nodes, or one of the nodes higher up in the hierarchy, provided there is a holistic token compatible with it. The choice between leaf nodes or their superordinate nodes is random, but we place an important restriction on the use of holistic tokens based on its information content. We will describe below how the agents keep track of all productions observed in memory. Using information content metrics extracted from this data, the agents are able to apply a word order for these sentences. The restriction on the use of holistic phrases is also based on these values: a holistic token can only be used if its information content⁹ is not significantly lower than the sum of the information content values of the names of its atomic concepts¹⁰.

The idea behind this is that it is only useful to employ a holistic token, if it does not overly generalize the situation and miss out on salient information. So if there is important information lost by using a holistic token, a compositional phrase of the atomic names is preferred. The basic intuition behind this is that if for instance **agent7** in the meaning structure in Figure 10.1 is very important to the state-of-affairs being expressed, a holistic utterance should not be preferred, as its general nature would cause **agent7** as informative entity to be disregarded.

There are two conflicting forces at work: the probability of holistic tokens will benefit from its general nature: a holistic token might therefore be more widely applicable. But this means its information content decreases the more frequently it is being used. The atomic concepts on the other hand can be

⁹As there is no sign of word order in the semantic structure yet, we measure its information content using a unigram model on the observed utterances.

¹⁰If the information content value of the holistic token is less than 90% of the value of the atomic token, the former's value is deemed to be significantly lower.

used less generally, but in a wider range of combinations, as there are no restrictions on its usage. Both atomic names and holistic names are driven by different forces that control their probability. The experiments will show which is the stronger.

Whether a holistic token is used to express a node in a semantic structure or a compositional one, does not have any effect on the rest of the production. The other nodes in the structure are expressed and conjoined with the token in the same way and holistic tokens are therefore treated as atomic objects from a syntactic point-of-view.

When there is more than one name to be expressed, the agent needs to decide on a way to juxtapose the names. In the experiments described in this chapter, we will concentrate on word order as the basic syntactic principle, but future research will include alternate methods for applying syntactic structure, such as inflections, prepositions, case, The initial word order in which the agents communicate the events is as good as random. We define six different possible ways to order subject (**S**), verb (**V**) and object (**O**) (if it is expressed)¹¹: **SVO**, **SOV**, **VSO**, **VOS**, **OSV** or **OVS**.

Initially, the agents have no preference at all for either order. In fact, different relationships can be expressed using different word orderings. Let us suppose the agent wants to express the fully compositional meaning of the mental structure in Figure 10.1. He therefore needs to express three different constructs, which each can be produced using a different ordering:

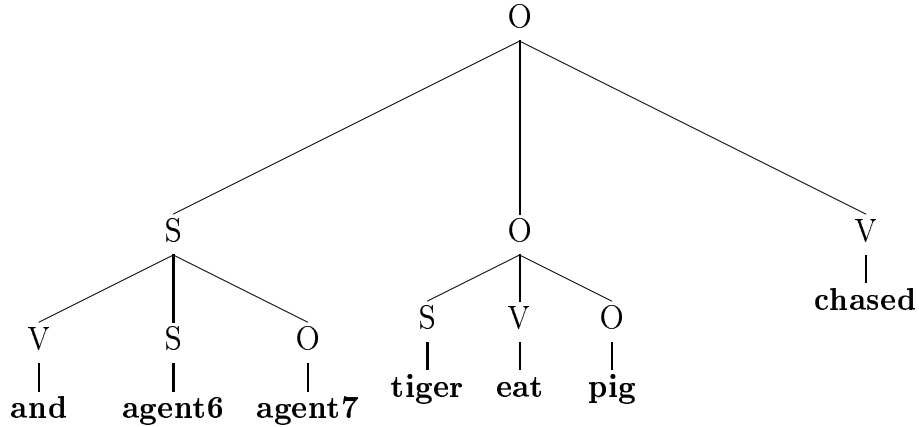
- chased(and,eat): **SOV**

- and(agent6,agent7): **VSO**

- eat(tiger,pig): **SVO**

So that finally the utterance itself can be produced:

¹¹These can be considered as placeholders. The verb is the relation/attribute being expressed, while the (syntactic) subject is the agent of the relation and its object the patient.



Thus, the utterance the agent uses to express the complex meaning is:

“And agent6 agent7 tiger eat pig chased.”

This method does not establish totally free word-order, as it groups mentalized groupings in the utterance, which seems an acceptable constraint on utterances. But for the other agent, it is impossible, at least initially, to tell what exactly is meant, as there are many different interpretations to this sentence and the referent, i.e. the actual unambiguous meaning being expressed, is not visible. He recognizes the basic objects and relations, but he can not unambiguously interpret this sentence.

The speaker will store the names he uttered in his lexicon. But if we are to suppose a basic cognition in the agent, it does not make sense to store explicit linguistic information in memory, in the form of rewrite rules and the like, as this presupposes a level of linguistic abstraction that the agents should not be assumed to have at this point. We do however consider the agents to have the ability to store frequencies of linguistic events. Given the string he produced, the agent records bigram frequencies as follows:

$\overset{1}{\curvearrowright}$ **and** $\overset{1}{\curvearrowright}$ **agent6** $\overset{0}{\curvearrowright}$ **agent7** $\overset{1}{\curvearrowright}$ **tiger** $\overset{1}{\curvearrowright}$ **eat** $\overset{1}{\curvearrowright}$ **pig** $\overset{0}{\curvearrowright}$ **chased**

So bigrams that are not directly headed by the same node, are not included in the count. Using this information, the speaker is able to reconstruct

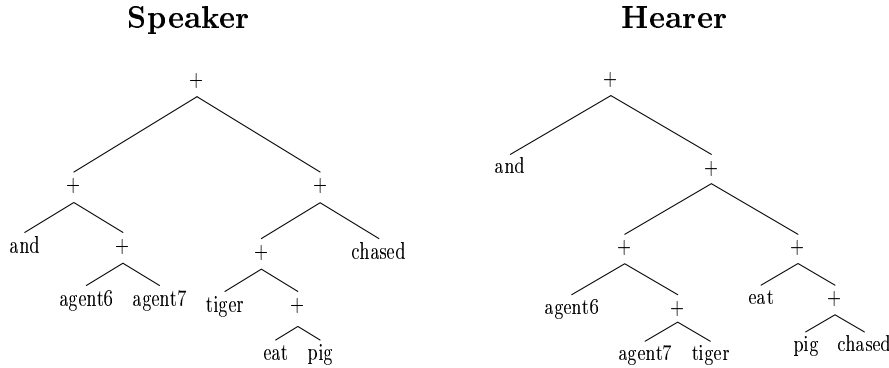


Figure 10.2: Agreement between speaker and hearer

the structure for his own sentence, using the unsupervised grammar induction technique described in Chapter 8.

In subsequent productions, the agent can use the bigram-probabilities recorded in memory to decide on a preferred ordering. For each concept to be expressed, the agent will consider the six aforementioned possibilities. A relation between an agent and a patient is expressed by at most three words. The triplet of words with the highest degree of lexical attraction on the bigram-level (cf. Chapter 8), will be the one the agent produces.

Let us now turn to the **hearer** agent. He observes this sentence, but has no access to its meaning. He therefore records the frequency of the bigrams, so that he can use these frequencies later to apply structure to new utterances. Without any prior knowledge, the hearer agent is still able to construct a structure for this sentence, albeit in a totally random fashion. We can then compare the structure of the hearer and the speaker to measure the degree to which they agree (cf. Figure 10.2). This information is however solely used as a way to measure how well the agents are agreeing, and is in no way used to guide communication, or provide some kind of fitness measure to the agents.

Upon hearing this utterance, the hearer agent will have some vague notion of what exactly the speaker is talking about. Naturally curious about the situation, the hearer agent will pick one of the objects from the utterance and repeat it to the speaker. The latter will then turn back to the matrix defined on page 351, find the object the hearer wants to know more about

and track a path to the end-state, for which he consequently will construct an utterance. The hearer will observe this utterance, again pick one of the objects he wants to know more about and so on and so forth.

In each language game, the hearer will ask 50 such questions, so that in the end, plenty of information about this scene has been conveyed. Even though the entire process seems long-winded the actual language game from a computational point-of-view is over in a matter of seconds.

This section introduced the basic cognition and conversational model that the agents employ in GRAEL-4. Let us now turn to the experiments that try to investigate the emergence of grammar using this system.

10.2 Experiments with GRAEL-4

In this section, we describe the general experimental setup and the course of the experiments themselves. As there are virtually no parameters to tweak, we can limit this discussion to the description of only one experiment, which was conducted three times, to compensate for the large degree of randomness involved.

10.2.1 Setup

A 10-agent GRAEL-4 society was initialized, in which the agents held no prior knowledge in their memory, apart from a lexicon, which we consider to have been acquired during the agent's early stages of language acquisition. Two agents are selected from the society, one speaker and one hearer and they play a language game as outlined above: the speaker recollects a scene and extracts a particular state-of-affairs from it. Then he presents a sentence expressing this scene which the hearer observes. The hearer picks one of the objects in the sentence, for which the speaker will present a new sentence based on his recollection of the scene. After the hearer has asked 50 questions, the language game is finished. The hearer will turn the speaker's last sentence into a semantic structure, that forms the basis of his own situation matrix.

Then, two agents are again chosen in the society to play a language

game with only one restriction: each agent will assume the role of speaker and hearer only once per language game run. When all agents have been a speaker and a hearer, the language game run is finished and the process starts anew.

To test the agreement of the agents, we record the F-score, based on the unlabeled precision and recall measures for the tree-structures as displayed in Figure 10.2. Since this does not necessarily tell us anything about how the agents converge on a grammatical system to express meaning, we also define a set of 100 structured meanings like the one in Figure 10.1 and have the agents create sentences for these meanings (without allowing them to record any distributional properties of their utterances). The output of the agents at this point provides the basis for a more qualitative appreciation of the degree of convergence of the grammatical systems that the agents employ.

The GRAEL-4 society is generation-based, but does not employ fitness functions, or the genetic transmission of information. This means that there is no need to consider a difference between splicing or crossover. At some points in the society new agents will be introduced, while other agents die at random intervals (lasting at least 200 runs, but without an upper-bound limit on their life-span). This allows for a dynamic population size, which is only restricted by imposing a lower-bound society size of 5 agents.

10.2.2 Experiments: Quantitative view

Figure 10.3 shows the average F-score in each language game run. For at least 1000 runs, nothing seems to be happening in terms of the F-score. The large degree of randomness does not seem to be replaced with any kind of systematicity. In fact, there even seems to be a slight decline. After 1000 runs, the situation slowly changes and an almost linear, but very subtle increase is clearly noticeable for at least 3000 runs. Between 3500 and 4000 runs, the linear increase evens out and F-scores start to settle down. Figure 10.3 bears some resemblance to the one we encountered in the unsupervised grammar induction experiment on the WSJ-corpus in Chapter (p. 298). The two other simultaneously conducted GRAEL-4 experiments each exhibited a plot that was more or less similar to the one in Figure 10.3.

Starting off with an average F-score of 35% and ending up with an F-

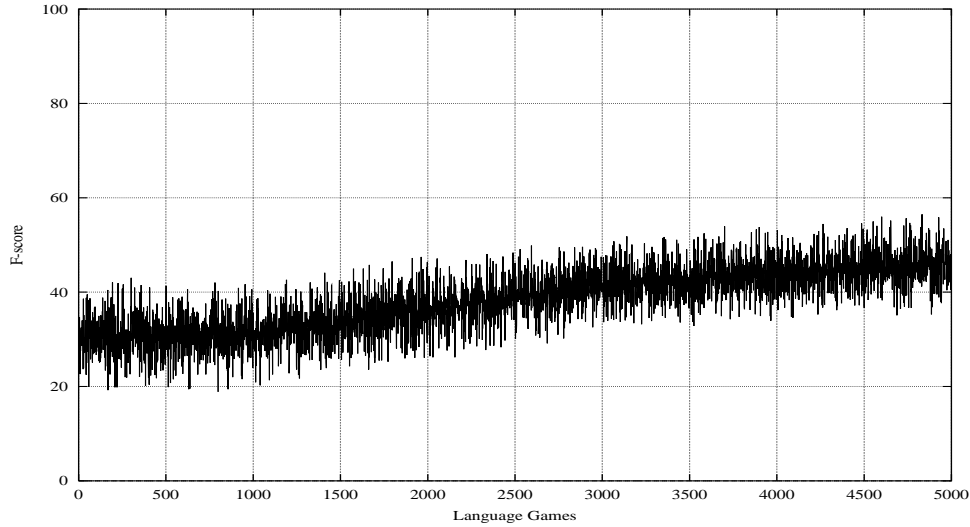


Figure 10.3: GRAEL-4 - Average F-score

score of 45% does not seem a great increase, but it is evident that some kind of change has occurred. It is indeed worth noting that the F-score is just an indicative score on tree-structures generated by a method that only to a limited extent is able to create fully-fledged syntactic structures. We have already pointed out in Chapter 8, that underwhelming F-scores do not necessarily mean that bad grammars have emerged/been induced.

The plot in Figure 10.3 indeed shows that some improvement is noticeable in the extent to which the agents agree in building syntactic structures for utterances. To really get a grasp of what exactly is happening in the GRAEL-4-society we need to conduct a more production-oriented data-analysis.

To check whether the agents are indeed developing some general notions on word order, we halt the society after every 1000 runs and extract the agents¹². We then place (the same set of) 100 pre-defined meanings in their mind, for which they need to render sentences. These 100 meaning constructs can be found in Appendix I. The degree to which the sentences that are produced are similar among the agents, expresses their convergence on a

¹²Given the dynamic population size, we sometimes needed to wait for at least 10 agents to become available. If there were more than 10 agents in the society, we randomly selected 10 agents.

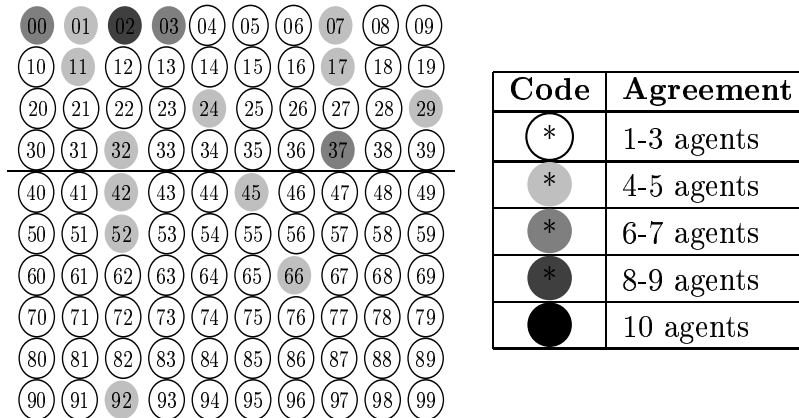


Figure 10.4: Convergence Diagram: Initial Situation

word order.

To illustrate this convergence, we use the diagram in Figure 10.4. This diagram displays the 100 meanings, starting off with the four attributes, followed by 36 simple relationships and followed by 60 complex meanings (cf. Appendix I). Figure 10.4 displays the situation at the start of the GRAEL-4 society. The four attributes (meanings expressed by at most two tokens) are trivially found by at least 1/2 of the population. The word order for some of the simple relations is also shared by several agents in some situations, either by a very lucky ordering of the words, or in the event of several agents choosing the same holistic utterance to express a complex meaning, reducing the randomness effect of word ordering.

Figure 10.3 showed that after 1000 language game runs, F-scores have not increased significantly throughout the society. This does not mean that nothing is being learned however as the convergence diagram on the left in Figure 10.5 illustrates: the word order for almost all attributes is shared by most all agents. At this point, the newborn agents' holistic languages are still the limiting factors for the observed convergence "scores". The simple relationships are also increasingly expressed with the same word order, even though the choice still seems largely random. The word order for meaning (17), which was shared by up to 5 agents in the initial stage (Figure 10.4) has now become randomized again, indicating that the initial convergence was a random effect, one that could not be maintained in the society.

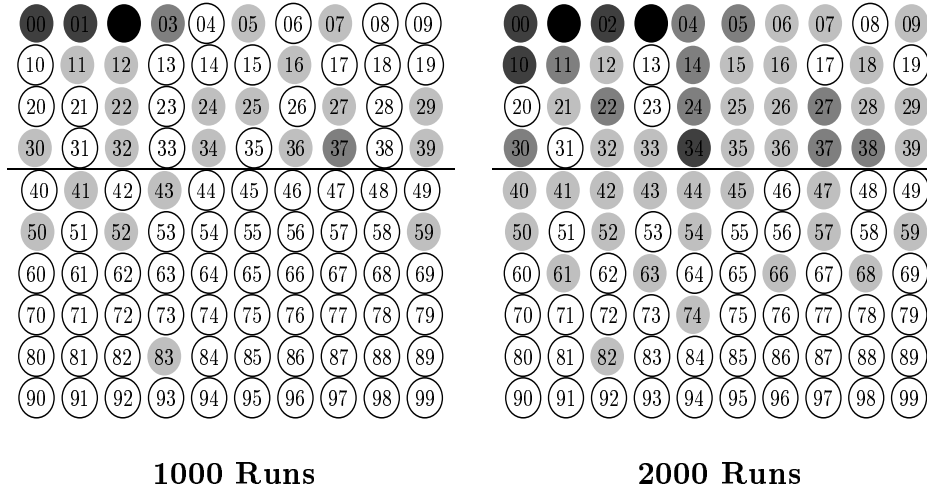


Figure 10.5: Convergence Diagram: 1000 Runs - 2000 Runs

There is however a difference noticeable between the meanings that express a relation between two [+animate] objects (meanings 4-21) and meanings that express a relationship with a [-animate] patient (meanings 22-39). On the whole, the word order seems less random for these meanings. It is trivial that agents have a much easier time, distinguishing word order between two classes of objects, whereas it is not transparent from the word order alone which object is the agent and which is the patient.

Figure 10.3 showed that after about 1000 runs, the F-scores start to increase linearly. Looking at the convergence diagram after 2000 runs (Figure 10.5, right-hand side), we notice that there does indeed seem to be a noticeable tendency for convergence. Almost all the simple relationships involving a [-animate] object are starting to be expressed by a majority of the agents using the same word order. The agents are also picking up on the other simple relationships.

The complex relationships are also catching up, even though there is still a very large degree of randomness noticeable. Note for instance how the convergence on meaning (82) disappears and convergence on meaning (83) arises. This is most likely due to the lower frequency of observations made where a relationship is the agent or patient of another relationship itself, so that the word order for these types of meanings is less easily learned.

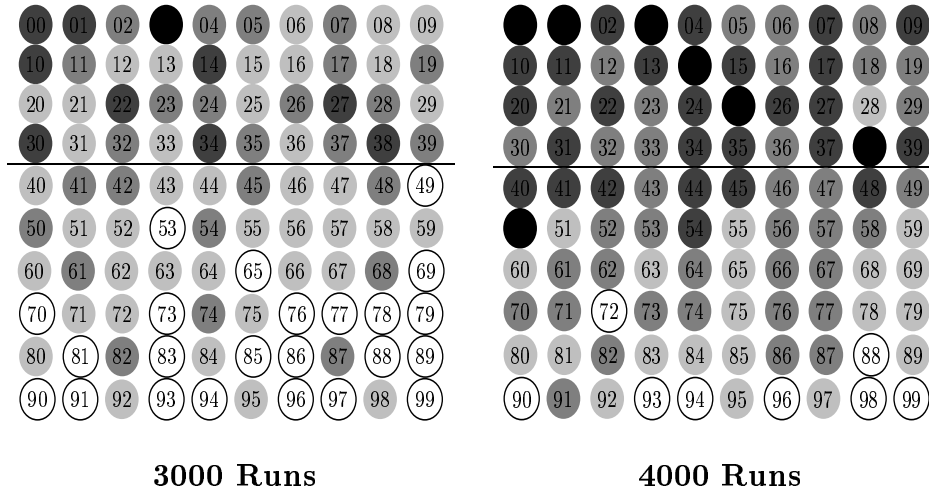


Figure 10.6: Convergence Diagram: 3000 Runs - 4000 Runs

Notice however the convergence on the first six complex meanings (40-45): these are simple meanings, in which one of the arguments is an attribute. The convergence on these meanings can be explained by the high degree of convergence noticeable on the single attribute meanings (1-4) itself.

The convergence diagram on the left in Figure 10.6 shows that this positive trend of convergence, also noticeable in Figure 10.3, continues. All simple meanings now display at least some limited degree of convergence, with many word orders shared by more than half of the population. It is mainly the young agents in the society that account for the randomness that is still present in the word ordering mechanism of the society. As convergence continues however, the newborn agents will be met with more consistent word order patterns, which helps them to pick up general tendencies more quickly. This is in line with Kirby's notion of the transmission bottleneck.

The right-hand side diagram in Figure 10.6 corroborates this claim. The situation has changed considerably, compared to 1000 games earlier: many complex meanings are now expressed in the same word order by more than half of the population. There is less disagreement on how to order sentences in which relations themselves are used as agent or patient, so that these patterns can be picked up faster by the newborn agents.

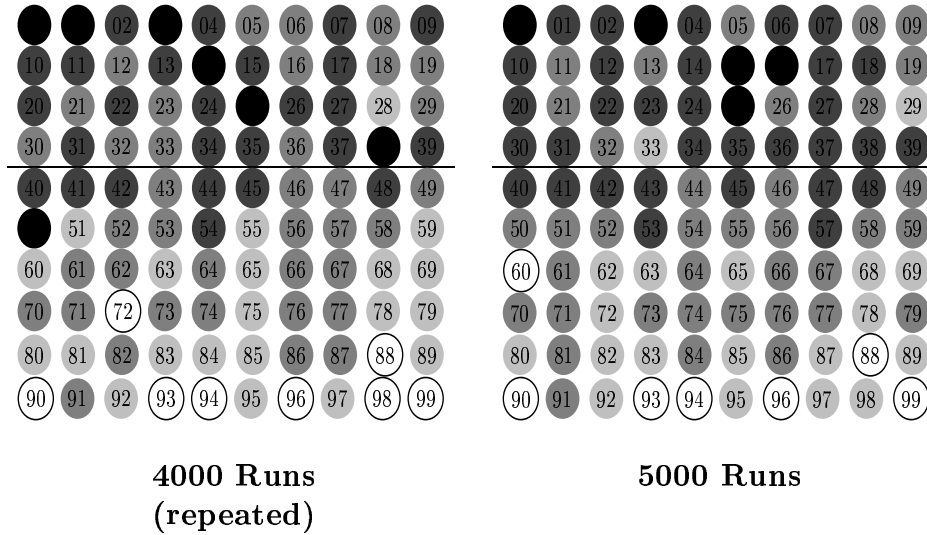


Figure 10.7: Convergence Diagram: 4000 Runs - 5000 Runs

At the end of the experiment (Right-hand side of Figure 10.7), nothing much has changed compared to the situation 1000 runs earlier. Convergence has improved on some meanings, while other (mostly complex) meanings seem to have become a bit more randomized (cf. (60)), even though this is probably due to newborn agents adjusting to the language at this point in time. No considerable increase has been gained in 1000 runs (also see Figure 10.3), indicating that either the society is residing at a local maximum, like it was during the first 1000 runs, or it has reached the degree of convergence that can be maximally expected from the GRAEL-4 society. The qualitative analysis in the next section indicates that the former interpretation holds true.

10.2.3 Experiments: Qualitative View

So far, we have presented a rather abstract view of the course of the GRAEL-4 experiment. Let us now look at the data in some more detail to see what exactly is going on. We will base our discussion on the previously presented meaning structure `chase(and(agent6,agent7),eat(tiger,pig))`, which was included in the set of 100 meanings (cf. meaning (91) in the convergence

diagrams).

At the onset of the society, word order seems totally random for this meaning. Many different variations for the example sentence can be observed, three of which are:

1. *“and agent6 agent7 tiger eat pig chased”*
2. *“eatanimateany agent7 and agent6 chased ”*
3. *“agent6 agent7 and eat pig tiger chased”*

Sentence (2) shows an example of a holistic token that was applied on the meaning substructure *eat(tiger,pig)*. Sentence (1) and (3) show a very different word ordering and given the fact that both agents tried to express the same meaning, we can consider them to be unintelligible to one another. They each convey the same concepts to one another, but it is for instance not clear who is chasing whom.

Looking at the situation after run 2000, we have noticed that many attributes are being expressed in the same word order already (Figure 10.5). This is of course not surprising, considering the fact that attribute concepts constitute a fairly limited set and only take [+animate] objects as their argument. Agents will observe sentences like *“pig and happy tiger food eat”* as well as sentences such as *“food happy tiger eat”* and infer that there is a stronger correlation between the words *“happy tiger”* than there is between *“food happy”*.

The rule of thumb, not only for attributes, but for relations as well, is that early observed word order is usually maintained throughout the rest of the society, even though gradual changes can occur. Our example sentence is seemingly still being generated with the same degree of randomness and it seems that the agents have not learned how to order relations yet.

After 3000 runs (cf. Figure 10.6), we do notice that the first constant word ordering is apparent among the agents. Again three samples from three agents:

1. *“and agent6 agent7 chased tiger eat pig”*
2. *“and agent7 agent6 chased pig eat tiger”*
3. *“and agent6 agent7 chased tiger pig eat”*

The agents seemed to have learned two things so far: *and* as a relation always takes front position, while *chased* is used between two arguments that instigate relations themselves. Holistic phrases at this point seem to have vanished from the society. The aforementioned conflicting forces between the information content of generally applicable, therefore probable holistic tokens and specific, unrestricted and thus also probable atomic tokens, has been settled in favor of the latter.

After 4000 runs, we have seen that almost all simple relations have a fixed word order. There are no general tendencies in terms of **SVO**, **SOV**, ... ordering. Instead each “verb” seems to introduce its own ordering. This ordering has been established on the basis of relationships in which the patient was a [-**animate**] object. These objects are indeed distinguishable from [+**animate**] objects in their distribution: they are less widely applicable and are therefore more salient in the observations. A strong lexical attraction between a [-**animate**] object and a verb, will result in their juxtaposition, which has an effect on the word order of the entire phrase and therefore on the observed lexical attraction between the verb and the [+**animate**] object. Even though there is a fundamental problematic issue at hand here, which we will discuss in Section 10.3, it is clear that a fixed word order has been established for simple relationships.

Figure 10.7 also shows that six agents have used the same word order to express our example meaning structure:

1. *“and agent6 agent7 chased eat tiger pig”*
2. *“and agent6 agent7 chased tiger pig eat”*
3. *“and agent6 agent7 chased tiger eat pig”*
4. *“and agent6 agent7 eatanimateany tiger chased”*

6 agents used word order (1), two agents used word order (2), one agent used word order (3) and a newborn agent (20 runs old) uses sentence (4). The latter’s “correct” word order for *and(agent6,agent7)* seems coincidental as it does not seem to have picked up any general notions of word order at this point.

1000 runs later, the situation is very similar. There are still only six agents who agree on the word order, but the remaining three have adopted

word order (2) above. The lexical attraction in the sequence *tiger pig* seems to be strong enough to keep them grouped together. Another agent had placed *chased* at the back of the sentence.

The qualitative data analysis shows that even though there is a large degree of randomness at work in GRAEL-4, fixed word order does emerge in the society. The fact however that the preferred word order for our example meaning hardly changes over 1000 runs (and five generations) indicates that this word order is here to stay.

But the randomness factor seems to allow for language evolution: if for some reason, the word order in sentence (2) above, which was already the one proposed by three different agents, takes over, language evolution occurs. This is a very slow process, driven to a large extent by a random factor. Further experiments will need to investigate whether extended processing will take the society out of this possibly local maximum and evolve into a society with an ultimately fixed word order, but it seems that this is the type of convergence that can maximally be expected to occur. The society seems instead to go from one local maximum of convergence to the next, which we will argue in Section 10.4 is a realistic view of language as a complex adaptive system.

10.2.4 Extra experiments

As we have mentioned before, the GRAEL-4 experiment was run three times. A similar situation developed in every experimental run: only the time-frame and the word order that evolved was different in each experiment. Holistic elements disappeared over time in all experiments but one: in one experiment, two holistic meanings had been maintained throughout the society on which the agents had converged. These meanings were:

```
happy([+animate])  
fear([+animate],any)
```

Analysis of the data did not reveal any reason as to why these particular phrases had been maintained and why this occurred in the same experimental run. We do not consider this as a problem for this system, as [Wray 1998] suggests holistic utterances are always present in natural language,

even today, as living fossils of early protolanguage.

The disappearance of holistic items in these experiments can be explained by the large amount of different meanings that can be expressed. This allows for high information content values of atomic elements, providing them with a reasonably high information content, that can easily overtake that of the holistic token. This effect in fact seems to simulate the abandonment of holistic utterances, as there is no need for them: everything can be expressed using a compositional sequence of atomic names with a higher information content value to boot.

We also conducted a single experiment in which one immortal agent was introduced into the GRAEL-4 society with a pre-defined and unalterable set of word orders. This agent had no noticeable effect on the development of the society.

10.3 Problematic Issues

Even though GRAEL-4 provided a computational model for the emergence of grammar in the form of “fixed” word order, there are some problematic issues surrounding the implementation, as well as our interpretation of the data. We have started this chapter by pointing out that with GRAEL-4, we are trying to define a computational model of the emergence of grammar that makes a minimal amount of assumptions on the cognitive abilities of its agents. GRAEL-4 achieves this by not presupposing any innate linguistic abilities in the agents. On the other hand, we do render the agents able to exploit a lexicon that is seemingly innate and universal for all agents. But this is an innateness assumption not even nativists dare to make.

We have avoided the issue by stating that the agents have developed a *naming insight*, during the early stages of language acquisition and/or emergence, that enables them to provide names for objects. [Smith 2001; Steels 1998b] show how we can develop such a lexicon in a model similar to GRAEL-4 using principles of co-evolution. We therefore consider GRAEL-4 to be a method that takes the accomplishments of this research and uses it as a given. We do however realize that this is a considerable shortcut, which may not agree with everybody’s view of grammar acquisition and emergence. Future

experiments might tackle this issue by incorporating a lexicon emergence stage in the GRAEL-4 model, as well as a method to simultaneously structure the meaning space, as exemplified in [Steels 1998b].

Another controversial issue may be the cognitive abilities the agents are attributed: despite a notable absence of linguistic abilities, the agents do seem able to make intricate computations on a large array of memorized linguistic observations. This seems to provide the agents with cognitive capacities that seem out of reach even for evolved humans. But the information theoretic approach is warranted if we do not consider these calculations as a mental reality, but rather as a reflection of the cognitive ability to memorize observations and the ability to remember the saliency of these observations. Future research should implement a mechanism that introduces noise in the agents' memory, to make the computation of these capacities less straightforward.

None of the methods that model the emergence of grammar, including GRAEL-4 described so far provide any insight into the distinction between comprehension and production. Language users are usually able to comprehend many more sentences than they are able to produce, indicating that there is some kind of mechanism that limits production. Future research should look into this matter and investigate whether agents in a GRAEL-4 have some implicit implementation of this mechanism, for example by using a preferred set of constructs.

Future research might also look into communicative attempts between adult agents and newborn agents and see if the development of the latter's language somehow mirrors that of child language acquisition. Typically, a child's language constitutes a simplified version of the adult's. Clearly this is not the case in GRAEL-4, as newborn agents' languages are allowed an equal amount of compositionality, with only the lack of a fixed word order as a limiting factor.

We introduced GRAEL-4 as an alternative computational method to model the emergence of grammar, whereas we mainly discussed the emergence of compositional language in Chapter 9. These are of course not one and the same thing. GRAEL-4 could be said to already start off with compositional language, but this may perhaps be overstating matters. Even though Kirby and Batali do indeed have their systems evolve from a holistic language to a

compositional language exhibiting grammatical properties, we have argued in Section 10.1.2 why this approach is paradoxical in intent. GRAEL-4 does allow for holistic utterances, but there is indeed a strong preference for the agents to prefer the atomic objects. Nevertheless, the random use of words in compositional sequences, as exhibited at the onset of a GRAEL-4 society, does not by itself constitute compositionality, as initially no meaning is being expressed in the way the atomic objects are joined. Similarly, there is no meaning in the way an early hominid constructs a holistic grunt from sounds to express some kind of meaning.

The point is however well taken that GRAEL-4's a priori predilection for compositionality and its usage of pre-defined atomic objects does not constitute a model in which the actual emergence of compositional language can be proved. We therefore think of GRAEL-4 rather as a model of the emergence of grammatical principles. In the experiment described in this chapter, we have modeled word order, but it would also be feasible to devise experiments that allow the agents to employ other syntactic tools.

The data analysis presented in the previous section however belies a more fundamental problem: we have seen how the agents are using the same word order to express the same meaning. We can therefore consider them to be able to understand each other in a conversation. If two agents are talking, it should indeed be clear to them, who is chasing whom and what animal is being eaten. But we cannot be sure that the word order in the phrase "*and agent6 agent7*" is indeed an expression of an internalized **VSO** word order in the agent's mind or simply the effect of distributional properties of the bigram "*and agent6*".

So even though we have evidence of a direct mapping between the agent's semantic structure and his syntactic structure, we cannot be sure that the syntactic structure is indeed expressing the proper (agent,patient) relationship, or if it is just a side-effect of beneficial bigram probability distributions. However, there is no reason to assume that there would be a stronger lexical attraction in the sequence "*and agent6*" than in the sequence "*and agent7*", so the choice should be more or less random. The word order is also further defined by the next bigram "*agent6 agent7*". Even though it is not transparent how exactly agent-patient relationships are expressed by the combination of the two bigrams, the fact of the matter remains that the majority of agents, each depending on a different set of linguistic observations have

learned to impose the same word order for this meaning. But even though we can consider the mapping from semantic structure to syntactic structure to be converged, we are faced with an empirical problem in that it is impossible to authenticate its validity. We hope that this appreciation of the data generated by GRAEL-4 can instigate a more general discussion on the signal-meaning relationship in the research field that implements computational models for the emergence of compositional language, as many of the reservations made in the discussion of the output of GRAEL-4, might also hold relevant for similar methods as well.

10.4 Concluding Remarks

In this chapter, we introduced GRAEL-4 as a computational model for the emergence of grammar. We have presented a first experiment with the model, in which the agents were provided with very basic cognitive mechanisms. More importantly, we tackled the mind-reading abilities that were attributed to the agents in Batali's and Kirby's systems. Whereas they viewed language as a means of communicating about visibly present situations, we presented language as a means of communicating mental constructs. As the agents have no immediately available referent, we had to make some important simplifications, most importantly with respect to the agents' lexicon. We were however able to show how the most typical and most expressive grammatical principle, i.e. word order, can emerge without direct reference to meaning, and without using specialized linguistic generalization techniques.

In doing so, we have mainly focused on *how* grammar emerges, rather than *why*. [Kirby and Hurford 2001] described how grammar emerges because of the transmission bottleneck, which forces language to become structured, while [Nowak and Jansen 2000] describes how compositional language emerged because of the need to express an increasingly complex set of meanings. GRAEL-4 did not implement such a driving force that compels the agent to develop a fixed word order. It merely develops as a side-effect of communication. This helps the agents to pick up fixed word orders more easily, but this does not translate into a selective advantage. Grammar can be considered to emerge, as a side effect of basic semantic constraints that translate into probabilistic distributional properties when communicating about them.

However, there is a large amount of work that still needs to be done to turn GRAEL-4 into a realistic computational model for the emergence of grammar. Future research should look into the emergence of lexical information, rather than predefining it at the onset of the society. Similar to our conclusion in Chapter 8, we would also like to check whether or not extending the scope of our statistical processing beyond that of the bigram, can lead to more intricate word order schemes. We would also like to provide a more elaborate world description. Right now, the distinction is limited to [+animate] and [-animate], but apart from that, anything goes from a syntactic, as well as a semantic point of view. Even though a pre-structured world, might provide an unfair bootstrap to the emergence of the grammar, it would constitute a realistic situation, as early hominids should also be considered to know that *eat(agent6,lake)* is not a possible situation. Providing more elaborate subcategorization properties should allow us to closely monitor the emergence of grammar over time and should help resolve the empirical problem we identified in Section 10.3.

One of GRAEL-4's seemingly biggest shortcoming is its apparent limited convergence on a word order. Especially considering the pre-defined lexicon, it may be surprising that the society does not converge on a fixed word-order more rigorously and faster than it does. But we would like to argue that this is in fact GRAEL-4's biggest asset. In Part II, we have observed many times that, from an engineering point of view, convergence results in lower results. The best results were mostly gained in the brief period of "beneficial confusion", before the society settles down into a converged language system. Most of the systems described in Chapter 9 are however primarily looking for this state of convergence. But language itself never converges and constantly adapts to a changing environment and seems to be driven by chaotic elements, introducing a large degree of randomness in language both from a synchronic, as well as a diachronic point of view.

Despite some of the unrealistic assumptions we made to bootstrap GRAEL-4, we consider its route of slow evolution from one local maximum to the next, a better approximation of natural language than finite convergence. Because even in a stable environment, language should be considered to evolve, simply because of the human mind, that is able to structure, and re-structure reality in an infinite number of ways. Viewing language as a way to communicate about states-of-affair that are not immediately present to the hearer, has

forced us to limit the scope of GRAEL-4, but does indeed provide us with the desired non-converging language model.

Another point of criticism that applies to most of the models described in Chapter 9 deals with language understanding from a computational, as well as a conversational point-of-view. There seems to be a view of natural language processing in these systems as a finite 1-to-1 mapping of meaning and syntactic structure. But even computational parsing systems processing huge amounts of language data (cf. Chapter 3) are not able to accomplish this task. Furthermore, not even human language users are able to provide such a 1-to-1 mapping: the same meaning can be expressed in so many different ways, many of which not only syntactic in nature, but heavily reliant on semantic and pragmatic principles as well. Why should we then expect the compositional languages that emerge in a computational model to not exhibit ambiguity of any kind, as well as allow for some margin of plausible misunderstanding in inter-agent communication? The models described in Chapter 9 can therefore be considered as too strict, in that they impose many less than plausible restrictions on language and its conversational properties, but also as too unrestricted, as they presuppose explicit linguistic abilities, unrealistic mind-reading skills, resulting in an a priori stilted transition from holistic language to compositional language.

We have argued that compositionality does not just constitute splicing holistic utterances in atomic objects, but also the manipulation of these objects to express meaning in itself. In this view, syntax is not just a way to generalize over data, but a linguistic module that can apply meaning to a string of words. GRAEL-4 showed that a general notion of word order emerges, expressing semantic agent-patient distinctions, simply on the basis of observed bigram statistics. In the same vein, we have shown in Chapter 8 how simple bigram statistics can build a full syntactic phrase structure. We therefore would like to contrast the nativist point of view that syntax is an innate capacity that pre-defines the way we are able to apply structure to the building blocks of our lexicon. We suggest that syntax emerges out of the probabilistic distributional properties of utterances formed on the basis of semantic structures we wish to express. In this view, syntax emerges as a side-effect, but then takes over to manage further production as an expressive language module in its own right. We have shown how GRAEL-4 models this evolution, using general cognitive mechanisms and without referring to

immediately available meaning constructs.

It's such a fine line between stupid, and clever.

David St. Hubbins - This is Spinal Tap - ©1984

11

Conclusion

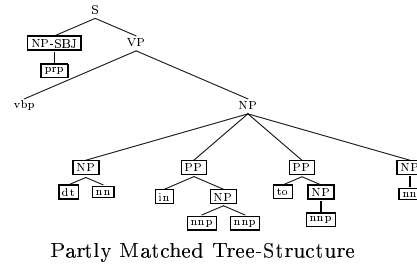
Throughout the different chapters of this thesis, we have covered a considerable range of research issues, extending from corpus-induced parsing to simulating the emergence of grammatical language. In this final chapter, we would like to overview the main insights and advances gained in these experiments (Section 11.1). It is clear that given the diversity of the tasks described in these pages, many unresolved issues are however still apparent, for which we will provide pointers to future research (Section 11.2). We will finish off by discussing, on a more general level, the merits of evolutionary computing in tackling NLP-problems pertaining to syntax (Section 11.3).

11.1 Advances

In Chapter 1, we attributed a *deconstructionist* intent to the course of experimentation throughout this thesis: we started off with a full specified corpus-induced parser, after which we introduced GRAEL as an agent-based evolutionary computing approach. By gradually dismissing each pre-defined

information source available to the society, we were able to tackle different NLP-tasks, such as grammar optimization and unsupervised grammar induction using roughly the same architecture. In this section, we will overview the results of the different experiments, ranging from the memory-based parser described in Chapter 3 to the simulation of the emergence of grammar in Chapter 10.

We started off with an overview of the machine learning method of memory-based learning, which has been successfully applied to a number of NLP-problems, but to a lesser extent full syntactic parsing. The Data-Oriented Parsing method however, is closely tied to the concepts that underpin memory-based learning. We therefore implemented a variant of Data-Oriented Parsing that magnifies the **pattern-matching** aspects of the system, by analyzing syntactic structures on the basis of substructures recorded in memory, or in other words: by looking at the *nearest neighbor* for a structure and consequently extrapolating its classification decision, similar to standard memory-based learning implementations like TIMBL that process propositional feature values.

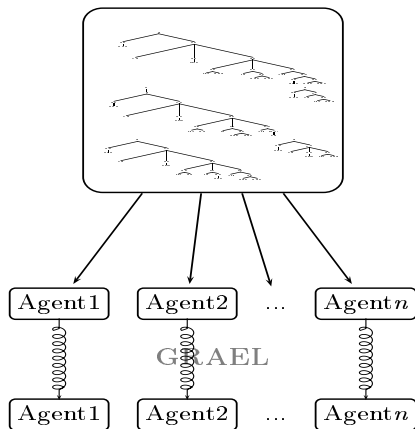


In the initial experiments, a standard PCFG was used to parse the test set, yielding less than optimal scores. We identified inherent limitations to the PCFG method, which causes flat structures to be preferred over structures featuring a large degree of embedding. The pattern-matching probabilistic grammar (PMPG) we devised, does exhibit a more robust approach to embedded structures. But error analysis shows that the PMPG greedily overestimates substructure size, yielding structures that are highly unlikely, yet made of large chunks of grammatical information recorded in memory.

A comparative data analysis between the output of the PCFG and the PMPG however showed that these systems complement each other to a certain extent. We therefore applied system combination, in which the PMPG was allowed to re-rank the parse forest proposed by the PCFG. This approach showed a significant improvement over each individual method and provides a workable memory-based corpus-induced parsing system, integrating the

common sense probabilities of the PCFG, as well as the context-sensitivity and memory-based aspects of the PMPG.

Despite the fact that the parsing systems described in Chapter 3 were able to exhibit scores that were in line with those reported by the state-of-the-art parsers, we did identify two major issues inherent to any statistical data-driven approach towards parsing: **probability mass distribution** and **grammar coverage**. Many sentences were provided with erroneous structures, but on inspection of the ordered parse forests, we found that the correct parse was very often to be found in the 10% most probable parses, leading us to assume that the probability-mass may not be optimally distributed over the elements in the grammar if we directly induce it from the training data. But even if we were to obtain an optimally distributed probability mass, parsing accuracy would still be hampered by another inherent limitation: suboptimal grammar coverage. Even on a small-scale corpus such as ATIS, many rules are not present in the corpus-induced grammar that are needed to parse the test set.



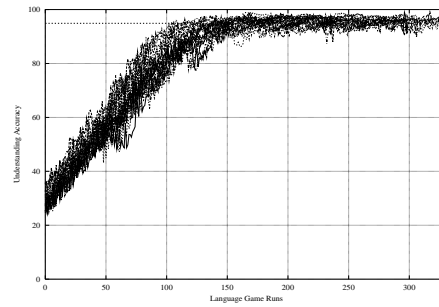
We therefore defined GRAEL: an agent-based evolutionary computing framework that can perform **grammar optimization** and **grammar induction** in an evolutionary environment that serves as a parallel optimization technique (distributed evolutionary computing) using a practical context (agents that *practice* the validity of their knowledge on each other). Typically, a corpus of tree-structures is distributed over agents in a GRAEL society. This leaves each of them with a small grammar that enables them to

process a limited set of sentences. The agents are then allowed to interact with one another in an extended series of so-called *language games*, the concept of which was borrowed from AI-research and adapted to the domain of data-driven grammar optimization. These language games entail that the agents parse each other's sentences, all the while helping each other out along the way by providing substructures of the correct solution. This enables the

agents to evaluate the validity of grammatical information in a practical context.

Chapter 5 described the first instantiation of the GRAEL environment, GRAEL-1, which leaves the structural properties of the original corpus-induced grammar intact. GRAEL-1 therefore functions as a **grammar optimization** technique, which can be said to redistribute the probability mass over the grammatical information originally induced from the annotated corpus. The new probability distribution therefore makes the grammar more suited for the task of parsing itself, rather than trying to reflect actual distributional properties of the training set.

We introduced a large array of different experimental parameters for the GRAEL-environment and proceeded to test them on the GRAEL-1 grammar optimization task. The experiments showed a clear increase in using GRAEL-1, indicating a beneficial redistribution of the probability mass. But the many different experimental parameters each affected scores to some extent: generation-based societies clearly outperformed single-epoch societies, which suggests that the use of generations and the associated fitness functions are able to weed out noisy and possibly harmful grammatical structures over time. It also provides a way out of local maxima, as the playing field is leveled every time an agent's grammar is partially transmitted to his offspring.



The fitness functions served as a way to provide the grammars with a particular bias: the most straightforward goal is to increase parse accuracy and fitness functions can be described that do indeed allow agents to evolve into genetically enhanced parsers. But it is also possible to allow for fitness functions that minimize grammar size, or maximize computational efficiency, if we require such a grammar from a practical point of view. Furthermore, a combination of fitness functions helps to counter overfitting effects, creating grammars that are more robust in dealing with new, unseen data.

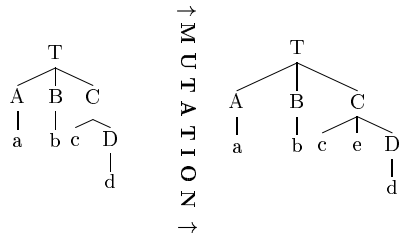
One of the most important issues however, was the **halting point**: given

that GRAEL allows for infinite processing, we need to find the best moment at which to halt the society and extract our grammar. Experiments showed that it is generally not a good idea to wait for a state of convergence to arise, as this means that the worst agents are getting better, but that the best agents are getting worse. We identified a brief window of time, occurring right before the aforementioned state-of-convergence, in which there is some *beneficial confusion* which yields the best agents in the entire lifespan of the society. It is at this time, that the probability mass is distributed over the agents explicitly on the basis of inter-agent interaction and that the convergence phase seems to return the distribution to that of the original training set itself.

The experiments with GRAEL-1 indeed showed that even without introducing new grammatical information, grammar optimization is possible in an agent-based evolutionary computing environment. While the use of agents makes sure the probabilities are enhanced in a practical context, the distributed nature provides a parallel processing technique in which several alternatives can be developed simultaneously¹. The evolutionary computing aspect of GRAEL-1 makes sure only those grammars survive over time that meet the fitness requirements we impose on the society.

¹The experimental parameter of society size can be adjusted to vary processing times and accommodate different data sizes.

Chapter 7 introduced *mutation* into the GRAEL environment, by considering a *noisy channel* in inter-agent communication, which causes elements in structures to be replaced, added or deleted. The source of the initial grammatical information is still the annotated corpus that was processed in GRAEL-1. We can therefore consider the grammatical elements to be pre-defined and also the labeling properties of the constituents, but the segmentation properties of said grammatical structures are abandoned. This projects GRAEL-2, as a grammar induction method, or more precisely a **grammar rule discovery** method, which can deal with the problematic issue of grammar sparseness identified in Chapter 3.



We performed a number of experiments, using the insights gained on GRAEL-1 to limit the combinations of experimental parameters to be considered. The most important adjustment we needed to make, was to allow the agents to use their I-language² to create structures for the sentences in their

E-language³, rather than maintaining the gold-standard parse from the original annotated corpus. This allows the agents to test the validity of the newly created grammatical information in a practical context.

Using a manually compiled worst-case scenario test set, we evaluated GRAEL-2 on the ATIS set and found it was indeed able to overcome the problematic grammar coverage issue. We were able to draw the same conclusion on the standard WSJ-test set, indicating that GRAEL-2 is indeed a workable grammar rule discovery method. A mutation operation such as the one used in GRAEL-2 renders many useless grammar rules, but the properties of GRAEL-2 allow the agents to evaluate the validity of mutated rules in a practical context, while the evolutionary computing aspects make sure that only useful grammar rules survive, as many different alternatives are being considered simultaneously.

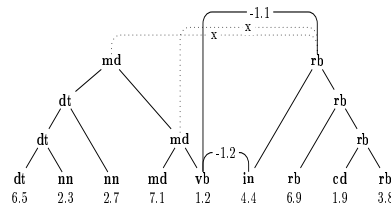
The experiments showed however that the grammar rule discovery functionality, caused GRAEL-2 to loose touch with the beneficial grammar opti-

²The set of grammar rules acquired during language games.

³The set of sentences that need to be parsed by other agents.

mization properties of GRAEL-1. We therefore initiated an experiment that started off with a GRAEL-2 society, but was turned into GRAEL-1 at a particular point in time to yield a grammar that has a large coverage, while at the same time featuring a probability mass that is optimally distributed. We believe that the combination of GRAEL-1 and GRAEL-2 constitutes an appropriate method to induce a large coverage grammar from an annotated corpus that can be used in a practical application that requires a good recall score, but not at the cost of precision.

Whereas GRAEL-2 can be perceived as a supervised grammar induction method as it creates variants of a corpus-induced grammar, GRAEL-3 does not require an annotated corpus and therefore constitutes an unsupervised approach to grammar induction. GRAEL-3 still needs a (large) amount

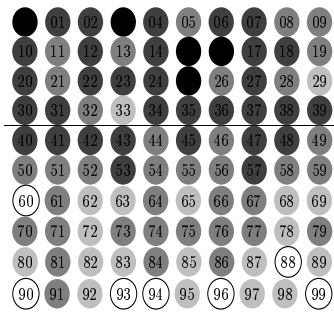


of sentences to be distributed over the society, but these sentences need not be pre-annotated. Chapter 8 described how we can use lexical attraction, a measure based on mutual information content, to bootstrap structure in the GRAEL-society. We introduced a simplified method that can apply structure to a sequence of words using lexical attraction in bigrams to provide the agents in GRAEL-3 with a basic notion of grammatical structure.

We presented two different approaches: GRAEL-3 can either take the entire collection of (unannotated) sentences, provide them with structures and distribute them over the agents, or it can distribute the sentences first and have the individual agents themselves apply structure. Since our grammar induction approach also constitutes a (very fast) parser, we are also presented with different methods for processing data in a GRAEL-3 society. Using the PMPG-parsing method defined in Chapter 3 in these experiments is problematic, as the ad hoc nature of the labeling properties of the grammar that is induced, has a detrimental effect on parsing accuracy. On the whole, the agents benefit from using the simplified parsing method as it produces better parse accuracies. But this approach does limit the beneficial aspects of GRAEL as an optimization method. Particularly on large data sets, grammatical information in the form of bigram probabilities is too limited as a knowledge source to yield considerable grammatical optimization by simple

re-distribution.

Even though GRAEL-3 only has a limited optimization effect on the original grammars, it does provide a significant improvement nevertheless. Using the same architecture for our grammar optimization task in GRAEL-1 and our grammar rule discovery method in GRAEL-2, we were able to devise an unsupervised grammar induction method, which can create grammars from scratch and consequently enhance and evaluate them in an evolutionary context. This presents GRAEL-3 as an interesting method to produce grammars for any kind of domain, provided there is enough textual data available.



Finally, we presented a computational model for the emergence of grammatical language. We provided an overview of the most important research efforts in this area in Chapter 9 and found that many of them provide the agents with a very strong linguistic bias and/or establish a communication model in which meaning is explicitly shared between participants. By further reducing the only pre-defined linguistic element that was left in GRAEL-3, i.e. the textual data, we can model the emergence of grammatical principles in a

society of agents.

We provided the agents with limited cognitive mechanisms that allowed them to name objects and record linguistic observations in memory. Language games occur when an agent recollects a scene and communicates about it to another agent. By using information culled from linguistic observations stored in memory, the agent can provide a sentence with a particular kind of word order. As the agents play more and more language games over time, the semantic properties of the objects that are expressed will enable the agents to apply a fixed word order on utterances.

Following the GRAEL-4 experiments, we suggested that grammar emerges, not as the result of an innate linguistic capacity, nor directly as a way to make language more learnable, but merely as a side-effect of the distributional properties of utterances formed on the basis of semantic structures we wish to express. A side-effect, which somewhere along the line took over as an

expressive module in its own right.

This section illustrated how we started off looking for a memory-based parsing system using pre-annotated data and ended up with a minimalist view of grammatical processing. The memory-based aspect can arguably be considered to be still present in the GRAEL-4, as natural language processing is still viewed as something that is done on the basis of previously recorded linguistic observations.

The problematic issues in PMPG have prompted us to consider an agent-based evolutionary computing approach to grammar optimization and induction, as this allows for the simultaneous development of several grammars, enhanced over time by evolutionary computing techniques implemented in a society of agents that interact in language games mirroring the task at hand: parsing unseen data. By reducing the predefined linguistic elements in GRAEL, we were consequently able to move from grammar optimization to supervised and unsupervised grammar induction to a computational simulation of the emergence of grammar, using the same computational architecture, but with different emphases and information sources.

11.2 Future Research

We have already identified many pointers to extended research in the respective chapters, some of which we will recap in this section. Furthermore, we would like to take a look at GRAEL from a more general point of view and propose some possible future research efforts that look into the validity of the approach, both from a computational point of view, as well as from a (psycho-)linguistic point of view.

The memory-based parser we described in Chapter 3 displays promising results on its own accord. The comparison with other systems however was problematic, as there were slight, but significant differences in the annotation properties of the data that was used during parsing. Future research that fine-tunes the performance of the PMPG should simplify the data in the same way as for instance [Collins 1999] does, to allow for a direct comparison with the state-of-the-art systems. To increase performance of the memory-

based parser, more time should be spent in the automatic optimization of the weights attributed to each classifier's decision towards parsing. Furthermore, different data sets (e.g. OVIS,...) should be used to test whether the enhanced memory-based aspects still hold for other types of tree-structures as well. This of course holds true for all data-driven GRAEL experiments as well: there should in principle be no restriction on the type of tree-structures that can be processed and consequently any type of grammar employing a tree-type structures, can be optimized by GRAEL.

The GRAEL-1 experiments constituted the bulk of the research in this thesis. When it comes to the investigation of GRAEL as a grammar optimization method, we consider most angles covered from an experimental point of view in Chapter 5. The largest amount of future work mostly pertains to computational issues: we were unable to conduct a standard GRAEL experiment on the WSJ-corpus because it would be too computationally expensive. Future research will need to compare the approximation of GRAEL used for the WSJ experiments, to the real deal. Perhaps a subsection of the WSJ corpus could be used to compare the approximation of GRAEL with the standard version, which will give us an insight that may translate to the experiment described in Chapter 5.

Based on the knowledge acquired during the GRAEL-1 experiments, we limited the amount of experiments on GRAEL-2. Future research should try to find out whether we missed some important details that can be beneficial to GRAEL-2. The use of a validation set may for instance provide an important performance boost as it has been observed to do for GRAEL-1. Also, the rather abrupt transition from GRAEL-2 to GRAEL-1 to provide an optimized replenished grammar is not very elegant. An integrated method should therefore be proposed that can perform grammar optimization, as well as grammar rule discovery in simultaneously.

We defined three different GRAEL-3 instantiations, but many more are possible: there are at least three different types of parsing going on⁴, and with at least two different possible parsers, we are presented with a wide range of possible GRAEL-3 methods. We also noted in the GRAEL-3 experiments that a very slow, but equally steady performance increase occurred. We should therefore revise our halting procedures and allow the slow progress to

⁴Language Game, Test Set Parsing, E-language Parsing,...

continue for a longer period of time.

We presented GRAEL-3 as a grammar induction method that can provide structure to any type of text. Even though many respectable structures can be found in a completely unsupervised manner, their results are underwhelming compared to a grammar induced from an annotated corpus. If our goal is to find a grammar that can be applied to a particular domain, we could however turn back to the combination of GRAEL-2 and GRAEL-1 as a supervised grammar induction method: after they have been allowed to process data of, for instance the WSJ-corpus, the agents could be provided with sample sentences of the domain for which we require a grammar. The agents can then use the grammatical information available to them at this point to adapt to the new domain over time.

GRAEL-4 constitutes arguably the most controversial set of experiments, as it claims to provide a minimalist approach to the computational simulation of the emergence of grammar, yet stipulates many assumptions on the agents that do not seem to provide a realistic situation. Future research will look into the emergence of the problematic lexical abilities the agents currently exhibit. This will also allow us to abandon the given nature of compositionality, and therefore resolve the initial distinction made between names and holistic utterances. Also, other grammatical properties such as intonation, case, prepositions, inflections,... need to be made available to the agents as possible syntactic tools. This will in particular allow us to view the emergence and evolution of grammar in terms of grammaticalization.

From a more theoretic point of view, we should look at child language acquisition and compare it to the GRAEL-4 approach. If “ontogeny epitomizes phylogeny” as [Studdert-Kennedy 1998; Wray 2000] suggest, there should indeed be parallels in the way the agents in GRAEL-4 learn grammar and children acquiring language. In the same vein, the grammar induction method of GRAEL-3 could be applied on a child language corpus like CHILDES to see if the structural aspects proposed by GRAEL-3 can predict the productions found in the data.

On a more general level, we could define a meta-GRAEL society in which we can allow agents from completely different societies to interact with one another. On a small scale, we can study what happens if we introduce WSJ agents to ATIS agents, but we can also construct a GRAEL-society with agents

from any of the four GRAEL instantiations. It will be interesting to see how fast for instance a GRAEL-4 agent adapts to a pre-structured environment and what effect it will have on the other agents. Likewise, we can combine agents that were trained on different language corpora and see if some kind of structured creole language arises over time.

Following the comparison between GRAEL and the ensemble learning methods of bagging and boosting in Chapter 6, we would also like to compare GRAEL to **active learning**. This technique has been identified in [Banko and Brill 2001] as a way to provide more annotated data with a minimal amount of human effort. In active learning, a trained set of learners classifies a set of (unannotated) data items. The instances that provide the most confusion among the set of learners, can then be considered the most useful for inclusion in the training data. Human annotators should then concentrate on these instances, while the machine learning algorithms take care of the unambiguous instances. [Banko and Brill 2001] describe a bagging approach to active learning, but it should also be possible to apply active learning within the GRAEL environment. After allowing the society to develop on an annotated corpus, the E-languages of the agents can be replaced by sentences from unannotated data. The confusion on these sentences can then be measured in GRAEL in a straightforward manner by looking at understanding accuracy scores during language games.

A final proposal for future research applies to the GRAEL method as a distributed evolutionary computing approach for NLP. The data-driven GRAEL experiments described in this thesis were very much applied to syntactic processing, but it would be interesting to see if we can take the basic sensibilities of GRAEL and apply them to other domains as well. Agents in the GRAEL-1 experiments for instance communicated by parsing each other's sentences using the memory-based parser defined in Chapter 3 and a set of grammatical constructs. But we could just as well give these agents a propositional learner as TIMBL and a set of feature values, which they can use to classify each other's data. Knowledge sharing can also be performed along similar lines. It would be interesting to see if GRAEL translates well to the propositional domain and other NLP-problems in particular.

11.3 Concluding Remarks

This thesis has presented one of the first research efforts that introduces agent-based evolutionary computing as a machine learning method for data-driven grammar optimization and induction. In recent years, many researchers have employed ensemble methods to overcome any negative bias their training data might impose on their classifiers. It is indeed important to view (annotated) data, not as an ideally distributed set of examples but as raw material that needs to be pre-processed before it can be used by a machine learning classifier. The bagging and boosting approach for instance, tries to create resamplings of the original data, to overcome the local maxima the data might restrict the classifier to (cf. Chapter 6), but we believe GRAEL adds an extra dimension to the task: by splicing the data and incorporating it in a society of communicating agents, we allow for the parallel development of several grammars at once, enhanced in a practical context that mirrors the ultimate goal: parsing unseen data.

We have shown how different parameters and information sources can help us tackle a considerable number of tasks pertaining to the development of grammar. With GRAEL we therefore believe to have developed a general framework in which grammars can interact and co-evolve. Rather than viewing this as a competitive situation, the agents mutually benefit from helping each other out, despite the fitness functions we may impose on them. And by lowering the plane at which evolution occurs from the genetic transmission of (grammatical) knowledge to the level of inter-agent communication, we have not only found a way to speed up evolution itself, but also to ground it in practical usage.

It is indeed the agent-based aspect of GRAEL that sets it apart from ensemble techniques such as bagging and boosting: rather than resampling the data, we provide agents with a partial solution which they have to built up in co-operation with each other. Rather than greedily trying to gather as much information as possible, the agents reconstruct data on the basis of experience on performing a particular task, e.g. parsing sentences. We therefore believe the redistribution of data to be the key feature to GRAEL, despite other types of functionality that can be integrated in GRAEL.

An important notion however is the degree of convergence in a GRAEL

society: experiments have shown that the biggest gain is to be found, not during the state of convergence, but in the state of what we have dubbed *beneficial confusion*. This is a direct consequence of the agent-based approach: convergence will occur if all agents possess a knowledge-base that is distributed along the same lines. With a high random factor evident in a GRAEL society, this kind of convergence can only reasonably be expected to occur when all agents have acquired a distribution of the data that starts to mirror that of the original training data. In other words: once convergence sets in, the beneficial redistribution GRAEL has provided is slowly being torn down in favor of a more conservative one. It is therefore important that the right parameters are set to (a) provide a large enough window of time for *beneficial confusion* and (b) halt the society at some point during that time.

In this vein, the parsing system we described in Chapter 3, can be interpreted as a GRAEL society with one agent, unable to play language games, causing an a priori and eternal state of convergence. GRAEL can then be considered as a method to break the convergence and move the grammar up to at least a higher plane of local maxima. The best grammars can then consequently be found right before the society reconverges.

And even with respect to GRAEL-4, in which there is no need from an engineering point of view to ever halt the society, we have argued why we do not wish to arrive at a state of convergence. A view of natural language as flowing from one local maximum to the next, not only helps to explain the diachronic evolution of language, but also the key aspects of language from a synchronic point of view: redundancy and ambiguity. These are indeed features in language that can be considered to constitute a local maximum state of language from a formal and functional point of view. But they are nevertheless features which make natural language what it is: a complex adaptive system in an ever-changing environment.

With its implementation of the “*divide and conquer*” principle, GRAEL provides a general framework for the development of grammars. Not only does it allow us to tackle engineering tasks like grammar optimization and induction, but it can also provide an initial insight into how early hominids might have developed grammatical language. GRAEL therefore establishes an environment to study the evolutionary dynamics of grammar itself. We believe grammar to be, not only the most productive module in natural language, but also the consequence of the basic capacity to create a men-

tal structure of a given situation. Even though we hardly give a moment's thought as to how this capacity allows us to transform complex thoughts into equally complex utterances, most people will attribute some degree of artificial intelligence to a computer program that can do the same, while others are prepared to pay thousands of dollars for the exhibition of grammar in an elephant wielding a paintbrush. If we indeed suppose that structure renders meaning and grammar governs structure, the investigation of the dynamics of grammar development seems paramount to establishing computational natural language understanding. We hope that the GRAEL system provides a way to initiate at least some part of the solution.

Bibliography

- Abney, S., R. Schapire, and Y. Singer (1999). Boosting applied to tagging and pp attachment.
- Adriaans, P. (1999). Learning shallow context-free languages under simple distributions. Volume 127. University of Stanford: CSLI-publications.
- Aha, D. W. (1997). Lazy learning: Special issue editorial. *Artificial Intelligence Review* 11, 7–10.
- Aha, D. W., D. Kibler, and M. Albert (1991). Instance-based learning algorithms. *Machine Learning* 6, 37–66.
- Allexandre, C. and A. Popescu-Belis (1998a). Emergence of grammatical conventions in an agent population using a simplified tree adjoining grammar. In *ICMAS98*, Paris, pp. 383–384. poster.
- Allexandre, C. and A. Popescu-Belis (1998b). Emergence of simplified tree adjoining grammar conventions in an agent population.
- Antonisse, H. J. (1991). A grammar-based genetic algorithm. In G. J. E. Rawlings (Ed.), *Foundations of genetic algorithms*, pp. 193–204. San Mateo: Morgan Kaufmann.
- Argamon, S., I. Dagan, and Y. Krymolowski (1998). A memory-based approach to learning shallow natural language patterns. In *Proc. of 36th annual meeting of the ACL*, Montreal, pp. 67–73.
- Argamon, S., I. Dagan, and Y. Krymolowski (1999). A memory-based approach to learning shallow natural language patterns. *Special Issue on Memory-Based Language Processing, Journal of Experimental and Theoretical Artificial Intelligence* 11(3).
- Baker, J. (1979). Trainable grammars for speech recognition. In *Proceedings of the 97th Meeting of the Acoustical Society of America*, pp. 547–

- 550.
- Baldwin, J. M. (1896). A new factor in evolution. *American Naturalist* 30, 441–451.
- Banko, M. and E. Brill (2001). Scaling to very very large corpora for natural language disambiguation. In *01*, Toulouse, France.
- Batali, J. (1998a). Computational simulations of the emergence of grammar. In M. S.-K. J. R. Hurford and C. Knight (Eds.), *Approaches to the Evolution of Language - Social and Cognitive Bases*. Cambridge: Cambridge University Press.
- Batali, J. (1998b). Computational simulations of the emergence of grammar. in approaches to the evolution of language: Social and cognitive bases. In M. S.-K. J.R. Hurford and C. K. (eds) (Eds.), *Approaches to the Evolution of Language: Social and Cognitive Bases*, pp. 405–426. Cambridge: Cambridge University Press.
- Batali, J. (2002). The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In T. Briscoe (Ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, Chapter 5. Cambridge University Press.
- Belz, A. (2001). Optimisation of corpus-derived probabilistic grammars. In *Proceedings of Corpus Linguistics 2001*, Lancaster University, UK, pp. 46–57.
- Bickerton, D. (1990). *Language and Species*.
- Black, E., R. Garside, and G. Leech (1993). *Statistically-driven computer grammars of English*. Amsterdam, The Netherlands: Rodopi.
- Blasband, M. (1998). Gag genetic algorithms for grammar. Technical report, Compuleer.
- Bledsoe, W. (1961). The use of biological concepts in the analytical study of systems. In *Proceedings of the ORSA-TIMS National Meeting*, San Francisco.
- Bod, R. (1995). Enriching linguistics with statistics: Performance models of natural language. Dissertation, ILLC, Universiteit van Amsterdam.
- Bod, R. (1996). Efficient algorithms for parsing the DOP model? A reply to joshua goodman. <http://xxx.lanl.gov/abs/cmp-lg/9605031>.

- Bod, R. (1998). *Beyond Grammar—An Experience-Based Theory of Language*. Cambridge, England: Cambridge University Press.
- Bod, R. (2000). Parsing with the shortest derivation. In *Proceedings of the 18th International Conference on Computational Linguistics*. Saarbruecken, Germany.
- Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of ACL-EACL 2001*.
- Box, G. (1957). Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society* 6(2).
- Breiman, L. (1996). Bagging predictors. *Machine Learning* 24(2).
- Brighton, H. (2002). Compositional syntax from cultural transmission. *Artificial Life* 8(1). eprint-ID: taal00000010.
- Brighton, H. and S. Kirby (2001a). Meaning space structure determines the stability of culturally evolved compositional language. Technical report, Language Evolution and Computation Research Unit, Department of Theoretical and Applied Linguistics, The University of Edinburgh.
- Brighton, H. and S. Kirby (2001b). The survival of the smallest: Stability conditions for the cultural evolution of compositional language. In J. Kelemen and P. Sosk (Eds.), *ECAL01*, pp. 592–601. Springer-Verlag.
- Brill, E. and P. Resnik (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. of 15th annual conference on Computational Linguistics*.
- Briscoe, E. (1998). Language as a complex adaptive system: co-evolution of language and of the language acquisition device. In *Proceedings of the 8th Meeting of Comp. Linguistics in the Netherlands*, Amsterdam, pp. pp. 3–40. Rodopi.
- Briscoe, E. (1999a). The acquisition of grammar in an evolving population of language agents. *Machine Intelligence, 16: Electronic Transactions in Artificial Intelligence, Special Issue*.
- Briscoe, E. (1999b). Grammatical acquisition: Co-evolution of language and the language acquisition device. In *Evolutionary perspectives on diachronic syntax*, *Proceedings of the Diachronic Generative Syntax*, Oxford. Oxford University Press.

- Briscoe, E. and J. Carroll (1993). Generalised probabilistic lr parsing for unification-based grammars. *Computational Linguistics* 19(1), 25–60.
- Briscoe, T. (1997). Co-evolution of language and of the language acquisition device. In P. R. Cohen and W. Wahlster (Eds.), *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, Somerset, New Jersey, pp. 418–427. Association for Computational Linguistics: Association for Computational Linguistics.
- Briscoe, T. and N. Waegner (1992). Robust stochastic parsing using the inside-outside algorithm. In *AAAI '92 Workshop on Probabilistically-Based Natural Language Processing Techniques*, pp. 39–53.
- Buchholz, S. (1998). Distinguishing complements from adjuncts using memory-based learning. In *Proceedings of the ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing, Saarbrücken, Germany*.
- Buchholz, S. and W. Daelemans (2001, September). Shapaqa: Shallow parsing for question answering on the world wide web.. In *Proceedings Euroconference Recent Advances in Natural Language Processing.*, pp. 47–51. Tsigov Chark, Bulgaria.
- Cardie, C. and D. Pierce (1998). Error-driven pruning of treebank grammars for base noun phrase identification. In *Proc. of 36th annual meeting of the ACL*, Montreal, pp. 218–224.
- Chandler, S. (1992). Are rules and modules really necessary for explaining language? *Journal of Psycholinguistic research* 22(6), 593–606.
- Chappelier, J.-C. and M. Rajman (1998, <ftp://ftp.inria.fr/INRIA/Projects/Atoll/TAPD98/chappelier.ps.gz>). A generalized cyk algorithm for parsing stochastic cfg. In *Proceedings of Tabulation in Parsing and Deduction (TAPD'98)*, Paris (FRANCE), pp. 133–137.
- Charniak, E. (1997, 27–31). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, Menlo Park, pp. 598–603. AAAI Press.

- Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*, Seattle, USA, pp. 132–139.
- Charniak, E., G. Carroll, J. Adcock, A. Cassandra, Y. Gotoh, J. Catz, M. Littman, and J. McCann (1996). Taggers for parsers. *Artificial Intelligence 85*, 45–57.
- Chen, S. (1995). Bayesian grammar induction for language modeling. *Proceedings of the association for computational linguistics*, 228–235.
- Chitrao, M. and R. Grishman (199). Statistical parsing of messages. In *Proceedings DARPA Speech and Language Workshop*.
- Chomsky, N. (1957). *Syntactic structures*. Den Haag: Mouton.
- Clark, A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In W. Daelemans and R. Zajac (Eds.), *Proceedings of CoNLL-2001*, Toulouse, France.
- Collins, M. (1997, july). Three generative, lexicalised models for statistical parsing. In *97*.
- Collins, M. (1999). *Head-driven Statistical Models for Natural Language Parsing*. Ph. D. thesis, University of Pennsylvania, Pennsylvania, USA.
- Collins, M. (2000a). Discriminative reranking for natural language parsing. In *Proceedings ICML-2000*, Stanford, Ca.
- Collins, M. (2000b). Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pp. 175–182. Morgan Kaufmann, San Francisco, CA.
- Cover, T. M. and P. E. Hart (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory 13*, 21–27.
- D. Mitchell, F. C. and M. Corley (1992). Statistical versus linguistic determinants of parsing bias: Cross-linguistic evidence. In *Fifth Annual CUNY Conference on Human Sentence Processing*, New York.
- Daelemans, W. (1999). Memory-based language processing. *Journal for Experimental and Theoretical Artificial Intelligence 11:3*, 287–467.
- Daelemans, W., P. Berck, and S. Gillis (1997). Data mining as a method for linguistic analysis: Dutch diminutives. *Folia Linguistica XXXI(1-2)*.

- Daelemans, W., S. Buchholz, and J. Veenstra (1999). Memory-based shallow parsing. In *Proceedings of CoNLL*, Bergen, Norway.
- Daelemans, W. and A. Van den Bosch (1992). Generalisation performance of backpropagation learning on a syllabification task. In M. F. J. Drossaers and A. Nijholt (Eds.), *Proc. of TWLT3: Connectionism and Natural Language Processing*, Enschede, pp. 27–37. Twente University.
- Daelemans, W., A. Van den Bosch, and A. Weijters (1997). iGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review* 11, 407–423.
- Daelemans, W., A. van den Bosch, and J. Zavrel (1999). Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*.
- Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch (1998). TiMBL: Tilburg Memory Based Learner, version 1.0, reference manual. Technical Report ILK-9803, ILK, Tilburg University.
- Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch (2000). TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, Tilburg University. available from <http://ilk.kub.nl>.
- Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch (2001). TiMBL: Tilburg memory based learner, version 4.0, reference guide. ILK Technical Report 01-04, Tilburg University. available from <http://ilk.kub.nl>.
- de Marcken, C. (1995). On the unsupervised induction of phrase-structure grammars. In *Proceedings of the Third Workshop on Very Large Corpora*.
- De Pauw, G. (2000a). Aspects of pattern-matching in dop. In *Proceedings of the 18th International Conference on Computational Linguistics*, pp. 236–242.
- De Pauw, G. (2000b). *Probabilistische Parsers - Contextgevoeligheid en Pattern-Matching*. Antwerpen, Belgium: Antwerp Papers in Linguistics.
- De Pauw, G. and W. Daelemans (2000). The role of algorithm bias vs information source in learning algorithms for morphosyntactic disam-

- biguation. In *Proceedings of the Fourth Conference on Computational Language Learning (CoNLL-2000)*, Lisbon, Portugal, pp. 19–24.
- Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning* 40(2), 1–22.
- Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10(7).
- Dik, S. C. (1997). *The Theory of Functional grammar. Part 1: The Structure of the Clause*. Berlin: Mouton de Gruyter.
- Dunbar, R. (1996). *Grooming, gossip and the evolution of language*. London: Faber.
- Dupont, P. (1994). Regular grammatical inference from positive and negative samples by genetic search: the gig method. In *Grammatical Inference and Applications*. Second International Colloquium ICGI-94: Springer Berlin.
- Fenk-Oczlon, G. (1989). Word frequency and word order in freezes. *Linguistics* 27, 517–556.
- Franz, A. (1996). Learning PP attachment from corpus statistics. In S. Wermter, E. Riloff, and G. Scheler (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Volume 1040 of *Lecture Notes in Artificial Intelligence*, pp. 188–202. New York: Springer-Verlag.
- Fraser, A. (1957). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science* 10.
- Freund, Y. and R. E. Shapire (1996). Experiments with a new boosting algorithm. In L. Saitta (Ed.), *Proceedings of the Thirteenth International Conference on Machine Learning*, San Francisco, CA, pp. 148–156. Morgan Kaufmann.
- Gillis, Steven, W. D. and G. Durieux (2000). Lazy learning: A comparison of natural and machine learning of stress. In P. Broeder and J. Murre (Eds.), *Models of Language Acquisition: inductive and deductive approaches*, pp. 76–99. Oxford University Press.
- Gold, E. M. (1967). Language identification in the limit. *Information and Control* 10, 447–474.

- Goodman, J. (1996). Efficient algorithms for parsing the dop model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 143–152. CEMNLP.
- Goodman, J. (1998). Parsing inside-out.
- Grunwald, P. (1994). A minimum description length approach to grammar inference. In S. W. G. Scheler and E. Riloff (Eds.), *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language*, Volume 1004 of *Lecture notes in AI*, Berlin, pp. 203–216. Springer Verlag.
- Harris, Z. (1951). *Methods in Structural Linguistics*. Chicago, IL: University of Chicago Press.
- Hashimoto, T. and T. Ikegami (1996). The emergence of a net-grammar in communicating agents. *Biosystems* 38, 1–14.
- Hemphill, C., J. Godfrey, and G. Doddington (1990). The atis spoken language systems pilot corpus.
- Henderson, J. and E. Brill (2000). Bagging and boosting a treebank parser.
- Henderson, J. C. and E. Brill (1999). Exploiting diversity for natural language processing. In *AAAI/IAAI*, pp. 1174.
- Honkela, T., V. Pulkki, and T. Kohonen (1995). Contextual relations of words in Grimm tales, analyzed by self-organizing map. In F. Fogelman-Soulié and P. Gallinari (Eds.), *Proc. ICANN'95, International Conference on Artificial Neural Networks*, Volume II, Nanterre, France, pp. 3–7. EC2.
- Hoste, V. and W. Daelemans (2000). Comparing bagging and boosting for natural language processing tasks: a typicality approach. In *Proceedings of the Tenth Belgian-Dutch Conference on Machine Learning (Benelearn 2000)*, Tilburg, The Netherlands, pp. 101–108.
- Huijsen, W.-O. (1993). Genetic grammatical inference: Induction of push-down automata and context-free grammars from examples using genetic algorithms. Master's thesis, Enschede, The Netherlands.
- Hurford, J. (2000). The emergence of syntax. In J. R. H. Chris Knight and M. Studdert-Kennedy (Eds.), *The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form*, pp. 219–230. Cambridge: Cambridge University Press.

- Jacoby, L. and L. Brooks (1994). Nonanalytic cognition: Memory, perception and concept learning. In G. Bower (Ed.), *Psychology of Learning and Motivation*, Volume 18, pp. 1–47. San Diego: Academic Press.
- Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics* 24(4), 613–632.
- Kammeyer, T. and R. Belew (1996). Stochastic context-free grammar induction with a genetic algorithm using local search. Technical Report CS96-476, Cognitive Computer Science Research Group, University of California, San Diego.
- Keller, B. and R. Lutz (1997a). Evolving stochastic context-free grammars from examples using a minimum description length principle. In *Workshop on Automata Induction, Grammatical Inference and Language Acquisition ICML-97*.
- Keller, B. and R. Lutz (1997b). learning stochastic context-free grammars from examples from corpora using a genetic algorithm. In *ICAN-NGA97*.
- Kirby, S. (1999). Syntax out of learning: The cultural evolution of structured communication in a population of induction algorithms. In D. Floreano, J.-D. Nicoud, and F. Mondada (Eds.), *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, Volume 1674 of *LNAI*, Berlin, pp. 694–703. Springer.
- Kirby, S. (2000). Syntax without natural selection: How compositionality emerges from vocabulary in a population of learners. In C. Knight (Ed.), *The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form*, pp. 303–323. Cambridge University Press.
- Kirby, S. (2001). Spontaneous evolution of linguistic structure: an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation* 5(2), 102–110.
- Kirby, S. (2002a). Learning, bottlenecks and the evolution of recursive syntax. In T. Briscoe (Ed.), *Linguistic Evolution through Language Acquisition: Formal and Computational Models*, Chapter 6. Cambridge University Press.
- Kirby, S. (2002b). Natural language from artificial life. *Artificial Life*. eprint-ID: taal00000037.

- Kirby, S. and J. Hurford (2001). The emergence of linguistic structure: An overview of the iterated learning model. In A. Cangelosi and D. Parisi (Eds.), *Simulating the Evolution of Language*, Chapter 6, pp. 121–148. London: Springer Verlag.
- Klein, D. and C. D. Manning (2001). Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the penn treebank. In *Proceedings of ACL-EACL 2001*.
- Koehn, P. (2002). Combining multiclass maximum entropy text classifiers with neural network voting.
- Kolodner, J. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann.
- Komarova, N. and M. Nowak (2001a). Population dynamics of grammar acquisition. In A. Cangelosi and D. Parisi (Eds.), *Simulating the Evolution of Language*, Chapter 7, pp. 149–164. London: Springer Verlag.
- Komarova, N. L., P. Niyogi, and M. A. Nowak (2001). The evolutionary dynamics of grammar acquisition. *Journal of Theoretical Biology* 209(1), 43–59.
- Komarova, N. L. and M. Nowak (2001b). Natural selection of the critical period for language acquisition. *Proceedings of The Royal Society of London. Series B, Biological Sciences* 268(1472), 1189–1196.
- Kool, A. (1999). Literature survey.
- Langacker, R. (1987). *Foundations of Cognitive Grammar*. Stanford, Ca: Stanford University Press.
- Langley, P. (1996). *Elements of machine learning*. San Mateo, CA: Morgan Kaufmann.
- Lankhorst, M. (1994). Breeding grammars: Grammatical inference with a genetic algorithm. Computer Science Report CS-R9401, University of Groningen, The Netherlands.
- Lari, K. and S. Young (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4(1), 35–56.
- Losee, R. (1995). Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. *Information Processing and Management*.

- Lucas, S. (1993). Biased chromosomes for grammatical inference. In *Proceedings of Natural Algorithms in Signal Processing*, Danbury Park, UK. IEE.
- Lucas, S. (1994). Context-free grammar evolution. In *First International Conference on Evolutionary Computing*.
- M. Redington, N. C. and S. Finch (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science* 22(4), 425–469.
- Magerman, D. and M. Marcus (1990). Parsing a natural language using mutual information statistics. In *Proc. of 8th. conference on AI (AAAI-90)*, Volume 2, pp. 984–989.
- Magerman, D. M. (1995). Statistical decision-tree models for parsing. In 95.
- Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger (1994). The penn treebank: Annotating predicate argument structure. In *Proceedings of ARPA Human Technology Workshop*, pp. 110–115.
- Marcus, M. P., B. Santorini, and M. Marcinkiewicz (1993). Building a large annotated corpus of english: the penn treebank. *Computational linguistics* 19, 313–330. Reprinted in Susan Armstrong, ed. 1994, *Using large corpora*, Cambridge, MA: MIT Press, 273–290.
- Matthews, P. H. (1997). *The Concise Oxford Dictionary of Linguistics*. Oxford: Oxford University Press.
- McLauchlan, M. (2001). Maximum entropy models and prepositional phrase ambiguity.
- Mel'čuk, I. A. (1988). *Dependency syntax : theory and practice*. SUNY Series in Linguistics. Albany: State University of New York Press.
- Mithen, S. (1996). *The prehistory of the mind*. London: Thames and Hudson.
- Mooney, R. J. (1996). Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 82–91.

- Nowak, M., J. P. and V. A. A. Jansen (2000, March). The evolution of syntactic communication. *Nature* 404, 495–498.
- Nowak, M., N. L. Komarova, and P. Niyogi (2001). Evolution of universal grammar. *Science* 291, 114–118.
- Pereira, F. and Y. Shabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proc. 29th annual meeting of the Association for computational linguistics*, Berkeley, California, pp. 128–135.
- Pinker, S. and P. Bloom (1990). Natural language and natural selection. *Behavioral and Brain Sciences* 13(4), 707–784.
- Plotkin, J. B. and M. Nowak (2000). Language evolution and information theory. *Journal of Theoretical Biology*, 147–159.
- Quine, W. V. O. (1960). *Word and Object*. Cambridge: MIT Press.
- Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Ramshaw, L. and M. Marcus (1995). Text chunking using transformation-based learning. In 95, pp. 82–94.
- Ratnaparkhi, A. (1997, June). A linear observed time statistical parser based on maximum entropy models. Technical Report cmp-lg/9706014, Computation and Language, <http://xxx.lanl.gov/list/cmp-lg/>.
- Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph. D. thesis, University of Pennsylvania.
- Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning* 34, 151–175.
- Ratnaparkhi, A., J. Reynar, and S. Roukos (1994, March). A maximum entropy model for prepositional phrase attachment. In *Workshop on Human Language Technology*, Plainsboro, NJ. ARPA.
- Rostand, E. (1897). *Cyrano De Bergerac*. Distribooks Inc.
- Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 898–904. AAAI Press.
- Roth, D. (1999). Memory based learning in NLP. Technical Report 2125, Urbana, Illinois.

- S. Buchholz, J. Veenstra, W. D. (1999). Cascaded grammatical relation assignment. In *Proceedings of EMNLP/VLC-99, University of Maryland, USA*.
- Scha, R. (1990). Taaltheorie en taaltechnologie: competence en performance. In Q. A. M. de Kort and G. L. J. Leerdam (Eds.), *Computertoepassingen in de Neerlandistiek*, LVVN-jaarboek, pp. 7–22. Almere: Landelijke Vereniging van Neerlandici. [Language theory and language technology: Competence and Performance] in Dutch.
- Scha, Bod, S. (1999). A memory-based model of syntactic analysis: data-oriented parsing. *Journal of Experimental and Theoretical Artificial Intelligence* 11:3, 409–440.
- Schapire, R. E. (1999). A brief introduction to boosting. In *IJCAI*, pp. 1401–1406.
- Schapire, R. E. and Y. Singer (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning* 39(2/3), 135–168.
- Schapire, R. E., Y. Singer, and A. Singhal (1998). Boosting and Rocchio applied to text filtering. In *In Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval, SIGIR '98*.
- Schoenemann, P. (1999). Syntax as an emergent characteristic of the evolution of semantic complexity. *Minds and Machines* 9, 309–346.
- Shannon, C. and W. Weaver (1949). *The Mathematical Theory of Communication*. Urbana, Illinois: University of Illinois Press.
- Sima'an, K. (1996). Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of the International Conference on Computational Linguistics (COLING '96)*, pp. pp.1175–1180 (vol. 2).
- Sima'an, K. (1999). *Learning Efficient Disambiguation*. Ph. D. thesis, University of Amsterdam.
- Sima'an, K., R. Bod, S. Krauwer, and R. Scha (1994). Efficient disambiguation by means of stochastic tree substitution grammars. In D. B. Jones and H. L. Somers (Eds.), *Proc. of the Int. Conf. on New Methods in Language Processing. Manchester, UK, 14–16 Sep 1994*, London, UK, pp. 50–58. UCL Press.

- Skousen, R. (1989). *Analogical modeling of language*. Dordrecht: Kluwer Academic Publishers.
- Sleator, D. and D. Temperley (1991). Parsing english with a link grammar. Technical report, CMU, Pittsburgh.
- Smith, A. D. (2001, September 10-14). Establishing communication systems without explicit meaning transmission. In J. Kelemen and P. Sosk (Eds.), *ECAL01, Lectures Notes in Computer Science*, Prague, pp. 381–390. Springer.
- Smith, T. C. and I. H. Witten (1996). Learning language using genetic algorithms. In S. Wermter, E. Riloff, and G. Scheler (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Volume 1040 of *LNAI*, pp. 132–145. Berlin: Springer Verlag.
- Spears, W. M., K. A. De Jong, T. Back, D. B. Fogel, and H. de Garis (1993). An overview of evolutionary computation. In P. Brazdil (Ed.), *Machine Learning: ECML-93, European Conference on Machine Learning, Vienna, Austria, April 5-7, 1993, Proceedings*, Volume 667 of *Lecture Notes in Computer Science*, pp. 442–459. Springer.
- Steels, L. (1997). Synthesising the origins of language and meaning using co-evolution, selforganisation and level formation. In J. Hurford (Ed.), *Evolution of Human Language*, Edinburgh. Edinburgh Univ. Press.
- Steels, L. (1998a). The origin of linguistic categories. In *The Evolution of Language Conference: Selected Papers from the 2nd international Conference on the Evolution of Language*.
- Steels, L. (1998b). The origins of syntax in visually grounded robotic agents. *Artificial Intelligence 103*(1-2), 133–156.
- Steels, L. (2000, August). The emergence of grammar in communicating autonomous robotic agents. In W. Horn (Ed.), *ECAI2000*, Amsterdam, pp. 764–769. IOS Press.
- Stolcke, A. and S. Omohundro (1994). Inducing probabilistic grammars by bayesian model merging. In *International Conference on Grammatical Inference*, unknown, pp. 999–999. unknown.
- Studdert-Kennedy, M. (1998). The particulate origins of language generativity: from syllable to gesture. In M. S.-K. J.R. Hurford and C. Knight

- (Eds.), *Approaches to the Evolution of Language: Social and Cognitive Bases*, pp. 202–221. Cambridge: Cambridge University Press.
- Tjong Kim Sang, E., W. Daelemans, W. D’ejean, H. Koeling, R. Krymolowski, Y. Punyakanok, and V. Roth (2000). Applying system combination to base noun phrase identification.
- Tjong Kim Sang, E. F. (2001). Transforming a chunker to a parser. In J. V. Walter Daelemans, Khalil Sima’an and J. Zavrel (Eds.), *Computational Linguistics in the Netherlands 2000*, pp. 177–188. Rodopi.
- Tjong Kim Sang, E. F. (2002). Memory-based shallow parsing. *Journal of Machine Learning Research* 2(Mar), 559–594.
- Tonkes, B. (2001, October). *On the Origins of Linguistic Structure: Computational models of the evolution of language*. Ph. D. thesis, University of Queensland, Australia. Draft PhD thesis, 140 pages.
- van den Bosch, A. and S. Buccholz (2002). Shallow parsing on the basis of words only: A case study. In *In Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL’02)*, Philadelphia.
- van Halteren, H., J. Zavrel, and W. Daelemans (1998, August 10-14). Improving data-driven wordclass tagging by system combination. In *98*, Montreal, Canada, pp. 491–497.
- van Halteren, Hans, J. Z. and W. Daelemans (2001). Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics* 27 (2), 199–230.
- van Rijsbergen, C. (1975). *Information Retrieval*. Butterworth.
- van Trijp, R. (2003). Modeling the evolution of syntax. Master’s thesis, Dept of Germanic Languages, University of Antwerp.
- van Zaanen, M. (2002, January). *Bootstrapping Structure into Language: Alignment-Based Learning*. Ph. D. thesis, University of Leeds, Leeds, UK.
- van Zaanen, M. and P. Adriaans (2001, October). Alignment-Based Learning versus EMILE: A comparison. In *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC); Amsterdam, the Netherlands*, pp. 315–322.

- Veenstra, J. B. (1998). Fast NP chunking using memory-based learning techniques. In *Proceedings of BENELEARN'98*, Wageningen, The Netherlands, pp. 71–78.
- Vilain, M. and D. Day (1996). Finite-state phrase parsing by rule sequences. In *96*.
- Wolff, J. (1998). Learning syntax and meanings through optimization and distributional analysis. In M. B. Y Schlesinger Levy (Ed.), *Categories and Processes in Language*, Lawrence Erlbaum, pp. 85–98. Hillsdale NJ.
- Wray, A. (1998). Protolanguage as a holistic system for social interaction. *Language and Communication* 18, 47–67.
- Wray, A. (2000). A protolanguage with no declaratives and no names. In *The Evolution of Language - 3rd International Conference: Proceedings*, Paris, France.
- Wyard, P. (1989). Representational issues for context-free grammar induction using genetic algorithms. Technical report, Natural Language Group, British Telecom, Ipswich UK.
- Wyard, P. (1991). Context-free grammar induction using genetic algorithms. In R. Belew and L. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo. ICGA: Morgan Kaufmann.
- Yang, C. (2000). *Knowledge and Learning in Natural Language*. Ph. D. thesis, Yale University: MIT.
- Yuret, D. (1998). *Discovery of Linguistic Relations Using Lexical Attraction*. Ph. D. thesis, MIT, Cambridge, MA.
- Zavrel, J. and W. Daelemans (1997). Memory-based learning: Using similarity for smoothing. In *Proc. of 35th annual meeting of the ACL*, Madrid.
- Zavrel, J., W. Daelemans, and J. Veenstra (1997). Resolving PP attachment ambiguities with memory-based learning. In M. Ellison (Ed.), *Proc. of the Workshop on Computational Language Learning (CoNLL'97)*, ACL, Madrid.
- Zhou, H. and J. Grefenstette (1986). Induction of finite automata by genetic algorithms. In *Proceedings of the 1986 IEEE International Con-*

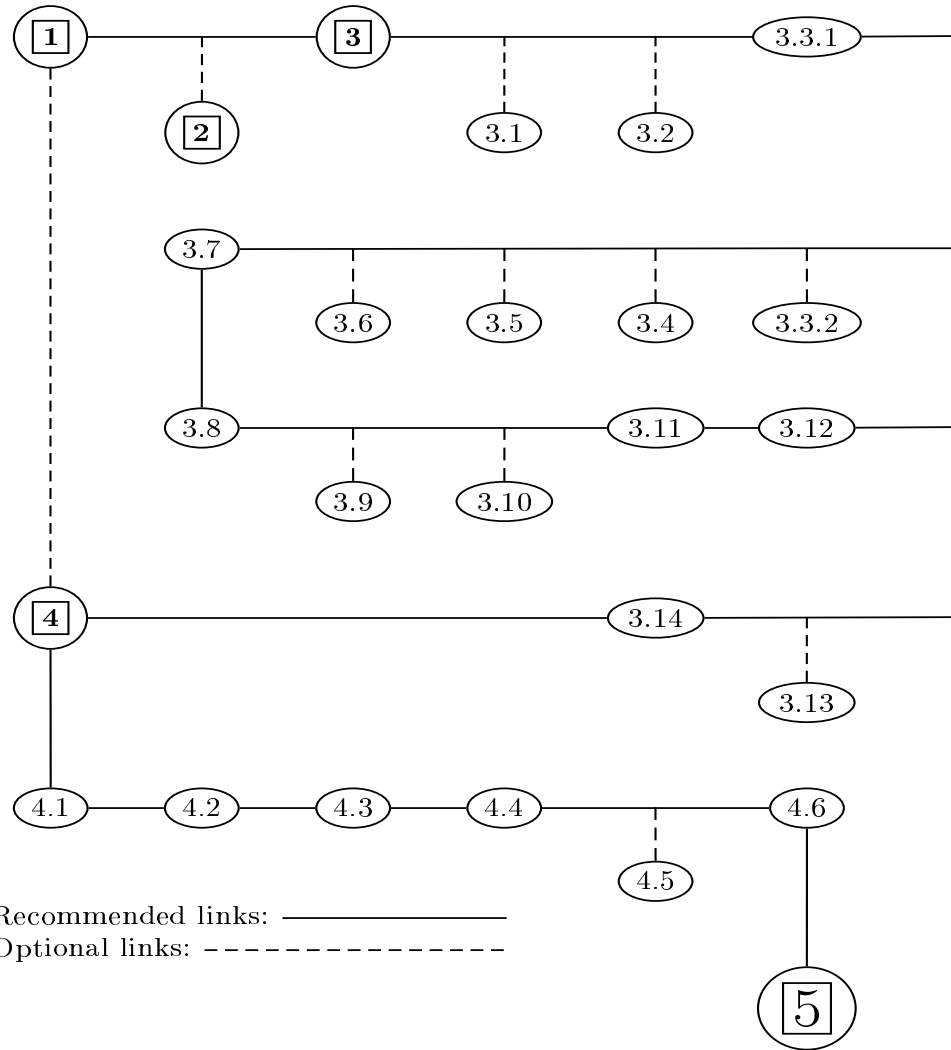
ference on Systems, Man and Cybernetics, Atlanta.

Zuidema, W. (2000). Evolution of syntax in groups of agents. Master's thesis, Theoretical Biology, Utrecht University.



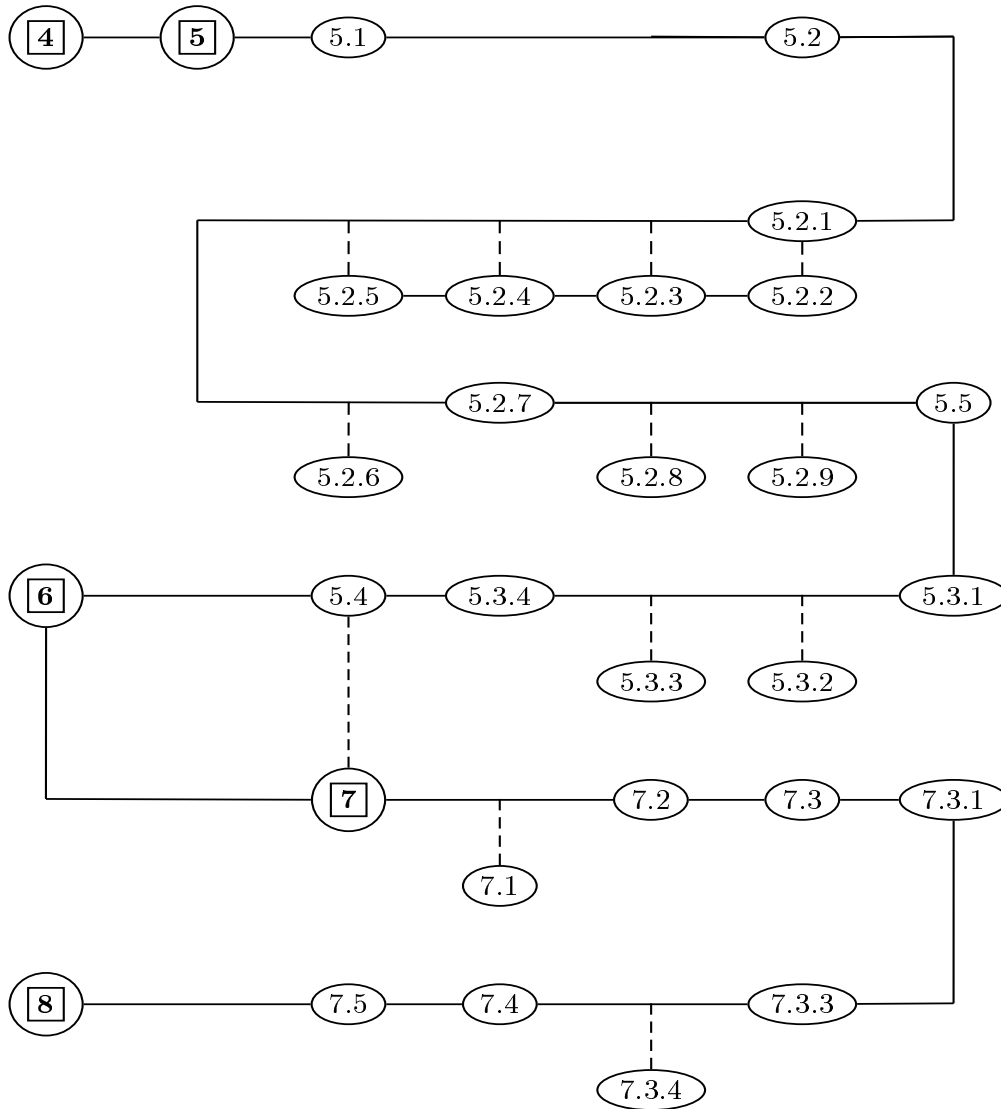
Paths through the chapters

Consult Tables A.1, A.2 and A.3 to plot a course through the chapters of this dissertation.



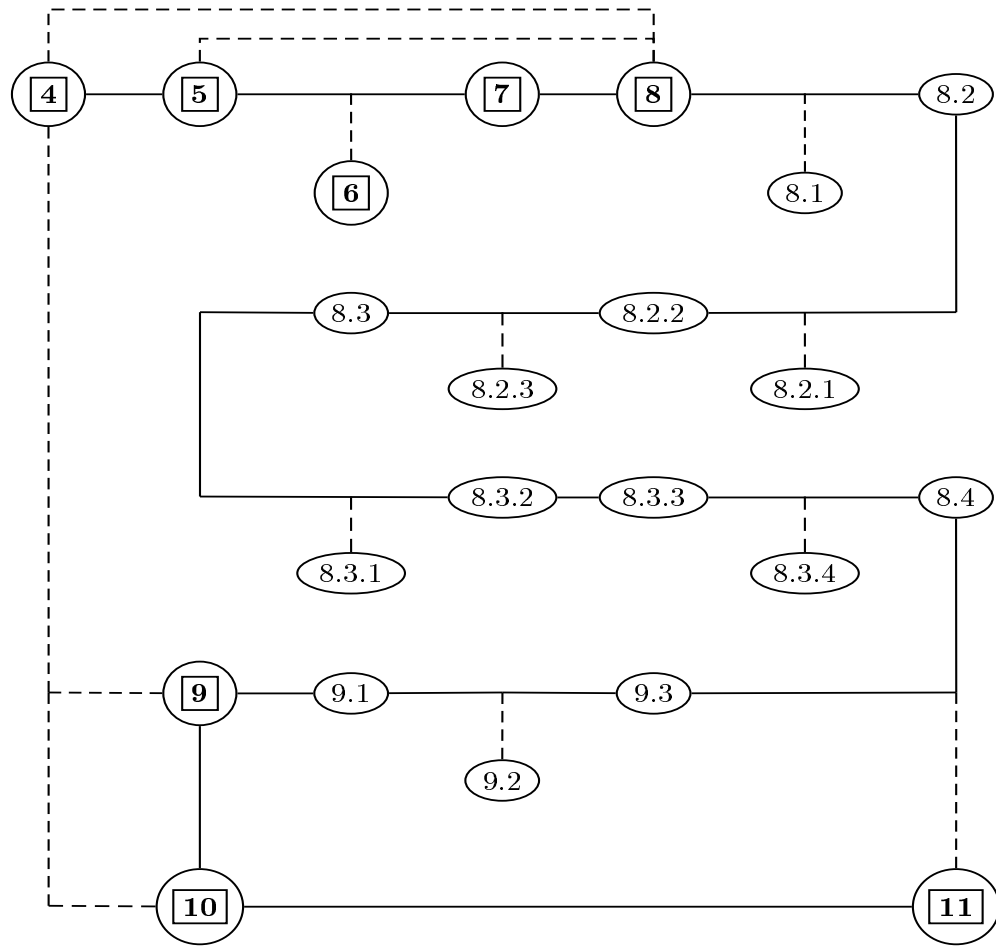
Many people will want to skip Chapter 2 entirely as it just provides a basic introduction into some concepts of machine learning and memory-based learning in particular. The reader can also skip Chapter 3 and go straight to Chapter 4, if (s)he is only interested in grammar optimization and induction itself, and not in the underlying data-driven parser. Whether or not to follow the other optional links inside the chapters, should be considered on the basis of how the titles relate to personal interest and background knowledge.

Table A.1: Path - Chapter 1 to 4



Chapter 6 can be skipped by people who are not interested in how GRAEL compares to other ensemble learning techniques. In principle, Chapter 7 can also be skipped if the reader is only interested in completely unsupervised grammar induction, rather than the intermediate approach GRAEL-2 provides. Subsections inside chapters can be skipped according to the reader's judgment

Table A.2: Path - Chapter 5 to 7



It is possible to skip the supervised optimization and induction methods of Chapters 5 to 7 and go straight to the unsupervised grammar induction method in Chapter 8, but readers should not do so without consulting Chapter 4. The data-driven experiments can be skipped altogether to advance to the experiment that investigate the emergence of compositional language. If the reader is knowledgeable about research in this area, Chapter 9 can be skipped as well. Chapters 9 and 10 can also be disregarded if the reader is only interested in engineering tasks, so that he can advance from the data-driven experiments in Chapter 8 to the conclusion in Chapter 11.

Table A.3: Path - Chapter 4 to 11

B

Complete Results for PCFG vs PMPG vs PCFG+PMPG experiments

B.1 ATIS

B.1.1 Labeled Precision

Partition	Absolute			%		
	PCFG	PMPG	PCFG PMPG	PCFG	PMPG	PCFG PMPG
1	420/499	446/522	451/499	84.2	85.4	90.4
2	360/398	384/460	390/411	90.5	83.5	94.9
3	396/436	410/502	428/466	90.8	81.7	91.8
4	437/493	451/500	468/501	88.6	90.2	93.4
5	360/402	435/480	445/460	89.6	90.6	96.7
6	360/440	366/456	391/430	81.8	80.3	90.9
7	454/515	461/531	467/510	88.2	86.8	91.6
8	365/425	384/449	402/429	85.9	85.5	93.7
9	441/493	391/482	448/481	89.5	81.1	93.1
10	355/387	382/418	381/413	91.7	91.4	92.3
Total stdev	3.948/4488	4.110/4800	4.271/4608	88.0 ±3.2	85.6 ±4.1	92.7 ±1.9

B.1.2 Labeled Recall

Partition	Absolute				%		
	PCFG	PMPG	PCFG PMPG	/const.	PCFG	PMPG	PCFG PMPG
1	420	446	451	496	84.7	89.9	90.9
2	360	384	390	468	76.9	82.1	83.3
3	396	410	428	479	82.7	85.6	89.4
4	437	451	468	512	85.4	88.1	91.4
5	360	435	445	485	74.2	89.7	91.8
9	360	366	391	471	76.4	77.7	83.0
7	454	461	467	519	87.5	88.8	90.0
8	365	384	402	446	81.8	86.1	90.1
6	441	391	448	490	90.0	79.8	91.4
10	355	382	381	453	78.4	84.3	84.1
Total	3948	4110	4271	4819	81.9	85.3	88.6
stdev					± 5.4	± 4.1	± 3.7

B.1.3 F-score

$F_{\beta=1}$ -score

PCFG	%	
	PMPG	PCFG PMPG
84.4	87.6	90.6
83.1	82.8	88.7
86.6	83.6	90.6
87.0	89.1	92.4
81.2	90.1	94.2
79.0	79.0	86.8
87.8	87.8	90.8
83.8	85.8	91.9
89.7	80.4	92.2
84.5	87.7	88.0
84.8	85.4	90.7
± 3.4	± 3.9	± 2.4

B.1.4 Exact Match

Partition	Absolute				%		
	PCFG	PMPG	PCFG PMPG	/sent	PCFG	PMPG	PCFG PMPG
1	32	36	39	57	56.1	63.2	68.4
2	39	44	48	58	67.2	75.9	82.8
3	30	41	45	58	51.7	70.7	77.6
4	40	45	49	58	69.0	77.6	84.5
5	35	36	37	58	69.3	62.1	63.8
6	42	42	46	58	72.4	72.4	79.3
7	33	40	43	58	56.9	70.0	74.1
8	36	36	42	58	62.1	62.1	72.4
9	26	35	36	58	44.8	60.3	62.1
10	34	40	43	57	59.6	70.2	75.4
Total stdev	347	395	428	578	60.0 ±8.3	68.3 ±6.1	74.0 ±7.5

B.2 Wall-Street Journal (10xv)

B.2.1 Labeled Precision

Labeled Precision

Partition	Absolute			%		
	PCFG	PMPG	PCFG PMPG	PCFG	PMPG	PCFG PMPG
1	2488/3683	2523/3683	3182/3642	67.6	68.5	87.4
2	2120/3156	2209/3156	2649/3117	67.2	70.0	85.0
3	2596/3966	2710/4112	3487/3995	65.5	65.9	87.3
4	2348/3562	2410/3667	2792/3588	65.9	65.7	77.8
5	2647/4025	2753/4173	3243/4040	65.8	66.0	80.3
6	2458/3621	2510/3767	2923/3607	67.9	66.6	81.0
7	2557/4881	2618/3994	3312/3835	52.4	65.5	86.4
8	2861/4266	2968/4427	3623/4297	67.1	67.0	84.3
9	2585/3926	2664/4100	3223/3918	65.8	65.0	82.3
10	2476/3707	2603/3707	3056/3711	66.8	70.2	82.3
Total Stevy	25136/38793	25968/38786	31490/37750	64.8 ±4.6	67.0 ±1.9	83.4 ±3.2

B.2.2 Labeled Recall

Labeled Recall

Partition	Absolute				%		
	PCFG	PMPG	PCFG PMPG	/const.	PCFG	PMPG	PCFG PMPG
1	2488	2523	3182	3780	65.8	66.7	84.2
2	2120	2209	2649	3325	63.8	66.4	79.7
3	2596	2710	3487	4189	62.0	64.7	83.2
4	2348	2410	2792	3598	65.3	67.0	77.6
5	2647	2753	3243	4094	64.7	67.2	79.2
6	2458	2510	2923	3693	66.6	68.0	79.1
7	2557	2618	3312	4019	63.6	65.1	82.4
8	2861	2968	3623	4545	62.9	65.3	79.7
9	2585	2664	3223	4061	63.7	65.6	79.4
10	2476	2603	3056	3794	65.3	68.6	80.5
Total	25136	25968	31490	39098	64.3	66.4	80.5
stdev					± 1.4	± 1.3	± 2.1

B.2.3 F-score

 $F_{\beta=1}$ -score

PCFG	%	
	PMPG	PCFG PMPG
66.7	67.6	85.8
65.5	68.2	82.3
63.7	65.3	85.2
65.6	66.3	77.7
65.2	66.6	79.7
67.2	67.3	80.0
57.5	65.3	84.4
64.9	66.1	81.9
64.7	65.3	80.8
66.0	69.4	81.4
64.5	66.7	81.9
± 2.7	± 1.4	± 2.6

B.2.4 Exact Match

Exact Match Accuracy

Partition	Absolute				%		
	PCFG	PMPG	PCFG PMPG	/sent	PCFG	PMPG	PCFG PMPG
1	7	21	26	192	3.6	10.9	13.5
2	16	27	28	192	8.3	14.1	14.6
3	13	16	22	192	6.8	8.3	11.5
4	19	24	37	192	9.9	12.5	19.3
5	18	25	31	192	9.4	13.0	16.1
6	21	30	33	192	10.9	15.6	17.2
7	16	23	26	192	8.3	12.0	13.5
8	15	24	28	192	7.8	12.5	14.6
9	20	26	30	192	10.4	13.5	15.6
10	18	20	26	193	9.3	10.4	13.5
Total	163	236	287	1921	8.5	12.3	14.9
stddev					±2.1	±2.0	±2.2

B.3 Wall-Street Journal (2.21/23)

	Absolute			%		
	PCFG	PMPG	PCFG PMPG	PCFG	PMPG	PCFG PMPG
Precision	33413/45905	30575/46684	37535/45908	72.8	65.5	81.8
Recall	33413/47333	30575/47333	37535/47333	70.6	64.6	79.3
Exact Match	265/2416	177/2416	386/2416	11.0	7.3	16.0



Correlation between experimental Parameters

We define the following variable experimental parameters:

Crossover	the situation in which crossover of grammatical structures occurs
Corpus	the annotated treebank used by the society
Halting Procedure	the method to determine when to halt the society and select its fittest agent
Fitness	the method to determine the fittest agent in the society
Generations	whether and how new generations are created
Population Size	the number of agents in a GRAEL society

Some of these parameters will necessarily have some influence on each other. The most straightforward example of such a correlation is the data set used and the number of agents in a GRAEL society. A large corpus may require a larger number of agents in a GRAEL society to achieve a good result. Another more

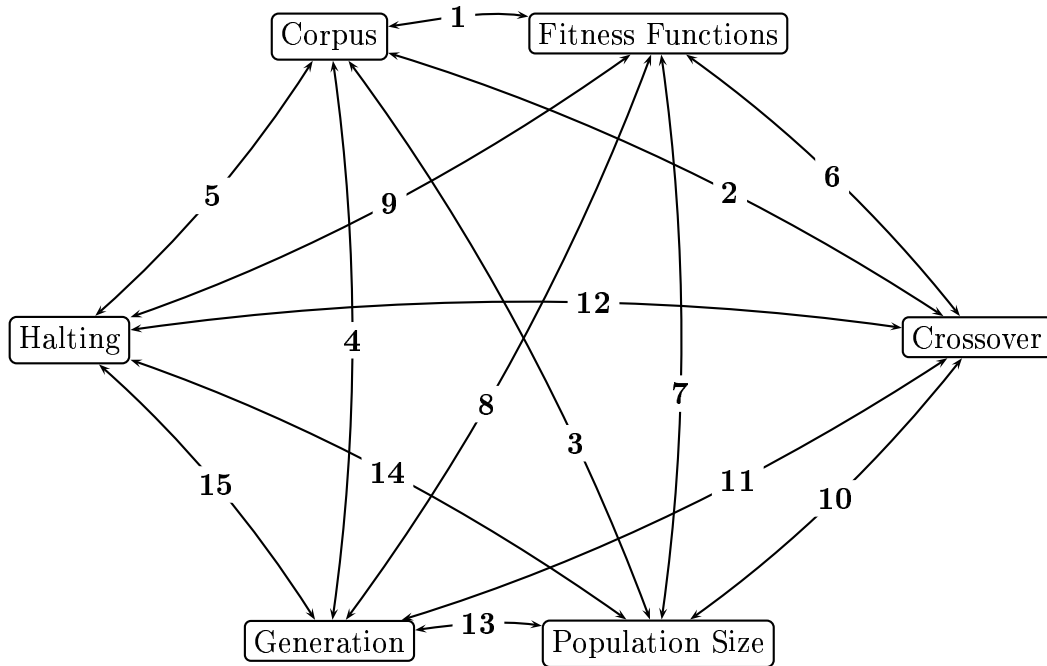


Figure C.1: Experimental parameters

complicated example involves the interdependence between the way in which new generations are created and the halting procedure. A halting procedure that looks at the accuracy of the full society compared to the baseline accuracy, may for example be dependent on the way new generations are created. But one might be less inclined to assume an interaction between the different types of crossover and the way new generations are created.

Figure C.1 displays all combinations of the experimental parameters defined in Section 5.1.2. For each combination, we will now consider whether or not it is reasonable to assume some kind of correlation between the two parameters that could have some influence on the performance of the society. First we will discuss those combinations of parameters that would seem to have an a-priori conspicuous correlation. This is followed by a discussion of less uncertain correlated parameters, after which we examine some combinations of parameters that could reasonably be considered to be uncorrelated. The numbers associated with the combinations, reflect those in Figure 5.1.2.

Strong Interdependence

1. Corpus \leftrightarrow Fitness Functions: fitness functions are paramount to the performance of any GRAEL society. But corpus-specific properties may influence the way in which the fitness functions operate in the GRAEL environment. The small and fairly homogeneous ATIS corpus may provide the agents with favorable scores for the fitness function of understanding throughout the society, rendering it less useful as a method for finding the fittest agent. And the *sparse grammar* problem apparent in previous experiments on the ATIS corpus may cause the agents to achieve unfavorable *accuracy* scores on the validation set, which may perhaps overemphasize it as a fitness function. It is clear that a corpus such as ATIS will need to employ different weights for the fitness functions than a large, heterogeneous corpus such as WSJ. The correlation between the corpus used and the optimal weights for the fitness functions should therefore be exhaustively investigated.

3. Corpus \leftrightarrow Population Size: the number of agents in the GRAEL society should have a direct influence on its performance. And whereas a homogeneous corpus may not need a large number of agents to provide a diverse distribution of grammatical information, it seems trivial that a larger scale corpus would need a larger society. The correlation between corpus and population size should therefore be researched.

7. Fitness Functions \leftrightarrow Population Size: the correlation of population size and the optimal combination of fitness functions is straightforward: the number of agents in a GRAEL society should have an effect all kinds of fitness functions, such as understanding accuracy, efficiency and size.

13. Population Size \leftrightarrow Creation of New Generations: when new generations are created by *end-of-life crossover* which joins the information of two agents to create new agents, the actual number of agents in a GRAEL society is an important factor in this process. A new generation in a small population may not differ very much from the previous generation, as combinatory possibilities between agents is by definition limited. The correlation between population size and the creation of new generations should therefore be exhaustively investigated.

15. Creation of New Generations \leftrightarrow Halting Procedure: the way in which generations are created may also be dependent on the halting procedure. Not only the limitation on the number of generations is evidently related to the way these generations are created, but also a halting proceeding like the relation between Full Society Accuracy and baseline accuracy is dependent on the constitution of a particular generation

Medium Interdependence

5. Corpus \leftrightarrow Halting Procedure: we defined several halting procedures, many of which are related to parsing accuracies on the data extracted from the annotated corpus, such as training set accuracy and parsing accuracy on a held-out validation set. The size and content of the actual corpus does not necessarily have to influence the way in which the halting procedure deals with these accuracies, but it can not be ruled out either. Since exhaustively investigating different halting procedure does not entail extra CPU-cycles (cf. *infra*), there is no reason not to check the correlation.

8. Fitness Functions \leftrightarrow Creation of New Generations: it does not seem apparent that the optimization of the fitness functions is strongly dependent on the way new generations are created. But since the fitness functions play an important role in the selection of agents for procreation, it seems advisable to exhaust the possible combinations anyway.

14. Population Size \leftrightarrow Halting Procedure: halting procedures such as the limitation on the number of language games or plateau detection may to a considerable degree depend on the size of the population. A smaller population will achieve convergence faster and therefore plateau sooner than a larger, more distributed society. The correlation between these two experimental parameters will therefore also be experimented on.

9. Fitness Functions \leftrightarrow Halting Procedure: the optimization of the fitness functions would not seem to be influenced by the way a GRAEL society is halted, or vice versa, but since the fitness functions themselves play an important role in some of the halting procedures, it is advisable to exhaust the combinations.

Weak Interdependence

10. Type of Crossover Operation \leftrightarrow Population Size: whether or not *random crossover* has any beneficial effects on the society can be related to its size. A larger population may be more robust in dealing with the random distribution of grammatical information random crossover entails, than a smaller society. The beneficial effect of random crossover may indeed depend on the population size and we should therefore experiment on the different combinations.

11. Type of Crossover Operation \leftrightarrow Creation of New Generations: the random crossover of grammatical information which happens when random

crossover is enabled, should have a noticeable effect on determining when an agent reaches end-of-life (cf. *infra*) which makes the two experimental parameters a priori related. Furthermore, random crossover is not computationally tractable in a Single Epoch GRAEL system, as it enlarges agents' grammars too fast, without the possibility to splice agents. And though it may not seem apparent that the way in which a new generation is created is dependent on the number of times it needs to be created, it seems advisable to check the combination between random crossover and the two methods for creating new generations.

No Interdependence

2. Corpus \leftrightarrow Type of Crossover Operation: intuitively random crossover would not seem able to yield a performance increase for the society as a whole. Random crossover only speeds up the process of the sharing of grammatical information. Language Game crossover on the other not only achieves the same goal (albeit at a slower rate), but also distributes grammatical information in a more intelligent manner. Any positive effect random crossover may have, can therefore also be achieved by lengthening the life-span of a GRAEL society through the halting procedure. Since checking for interdependence between corpus and crossover would effectively double the amount of experiments that need to be run, with little or no added insight to be gained, we only experiment on the crossover operation on one corpus, i.e. the ATIS corpus.

4. Corpus \leftrightarrow Creation of New Generations: whether or not new generations are created and the way in which new agents are introduced in new generations, is possible related to the fitness functions and population size. The nature of the grammatical content of the agents in a society would however intuitively not appear to play a vital role on the performance of a particular method of creating new generations. We therefore limit the experiments to exhaustive experiments for different methods on the ATIS corpus.

6. Fitness Functions \leftrightarrow Type of Crossover Operation: the performance of different fitness functions seems more dependent on the nature of the corpus and the overall architecture of the society (population size, ...) and less on another factor that has to do with inter-agent communication, such as the type of crossover employed throughout the society. It is not clear what insights are to be gained when we alternate fitness functions and crossover operations to check for correlation effects.

12. Type of Crossover Operation \leftrightarrow Halting Procedure: it is hard to

imagine a situation in which the addition of random crossover would have any effect on the best way to halt a GRAEL society. The rapid distribution of grammatical information will undoubtedly have an effect on various accuracy rates, but not on the actual method these halting procedures employ to determine the appropriate halting point.

Figure C.2 summarizes this thought exercise and adapts Figure C.1 to exclude those combinations of experimental parameters that intuitively seem unrelated. Note that the absence of experiments for these combinations is mainly because of reasons of computational tractability and that a definite answer about the correlation between these parameters is only possible in an experimental context, rather than the thought exercise we performed in this section.

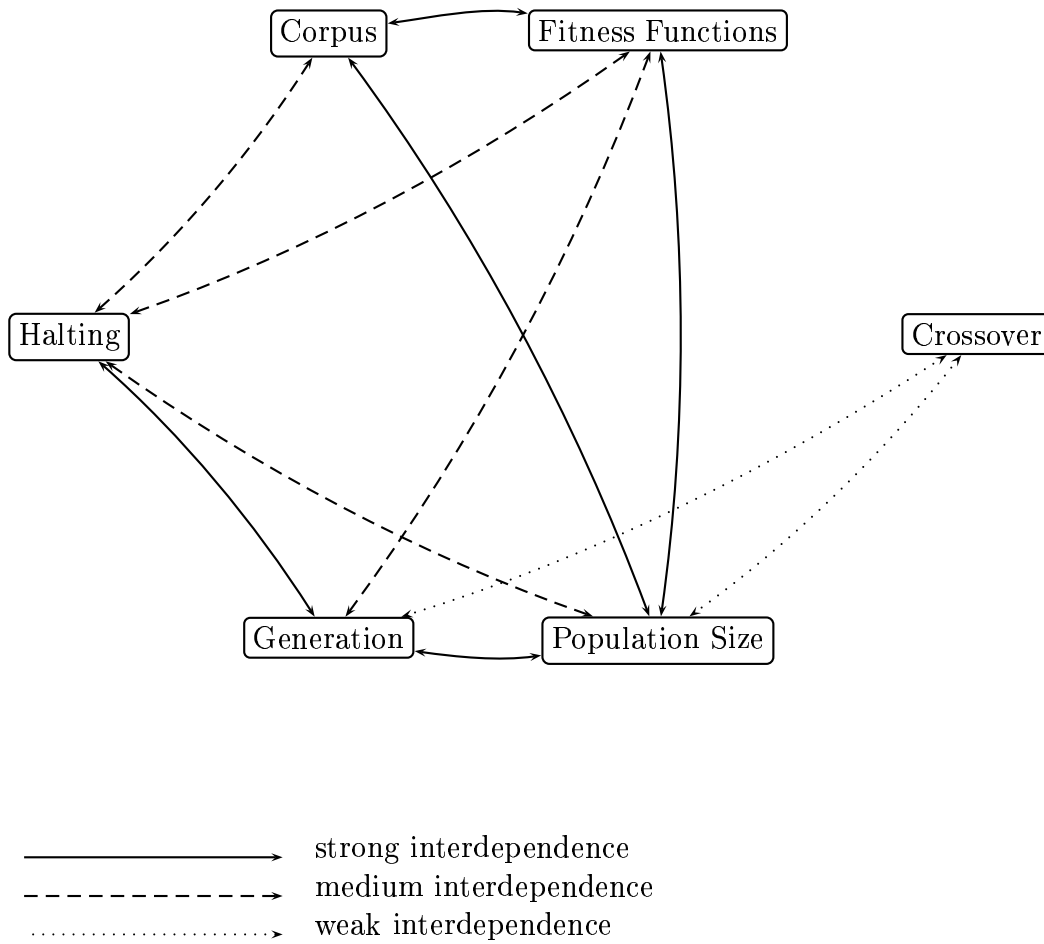


Figure C.2: Correlation of experimental parameters

D

GRAEL-1 ATIS Full Results Tables

D.1 20 Agents

D.2 10 Agents

D.3 5 Agents

D.4 50 Agents

D.5 100 Agents

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	90	80.1	—	—	—	—	115	89.9	146	89.9	200	90.2	—	—	200	90.2
Efficiency	90	88.2	106	90.8	—	—	104	90.7	146	90.9	200	90.9	—	—	146	90.9
Accuracy	90	84.2	131	91.1	131	91.1	131	91.1	146	91.0	200	91.0	—	—	131	91.1
Understanding	90	83.4	120	91.1	143	91.0	114	91.1	146	91.0	200	90.9	—	—	143	91.0
Understandability	90	78.6	138	90.3	—	—	120	87.3	146	90.7	200	90.8	—	—	146	90.7
Internal Consistency	90	72.3	—	—	—	—	149	90.6	146	90.7	200	90.8	—	—	200	90.8
Efficiency & Size + Accuracy	90	75.4	143	91.1	143	91.1	143	91.1	146	91.1	200	91.1	—	—	143	91.1
US&UB + Accuracy	90	89.9	98	90.7	107	91.1	98	90.7	146	91.0	200	91.1	—	—	107	91.1
Efficiency & Size + US&UB	90	82.4	131	91.1	137	91.1	130	91.0	146	91.1	200	91.1	—	—	137	91.1
Efficiency & Size + Accuracy + US&UB	90	80.4	136	91.1	137	91.1	136	91.1	146	91.0	200	91.1	—	—	137	91.1

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.1: GRAEL-1 ATIS- 20 Agents - Single Epoch - Halting Points and F-scores on Test Set

Table D.2: GRAEL-1 ATIS - 20 Agents - Splicing - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	—	—	203	76.9	250	80.3	381	78.5	381	78.5
Size (10%)	—	—	—	—	—	—	—	—	182	83.6	250	86.6	381	82.3	381	82.3
Efficiency	—	—	—	—	—	—	—	—	188	86.3	250	89.1	353	85.9	353	85.9
Accuracy	180	91.8	235	92.0	235	92.0	81	63.3	181	91.8	250	91.9	341	91.7	235	92.0
Understanding	209	91.8	186	91.7	186	91.7	186	91.7	176	91.5	250	91.8	370	91.1	186	91.7
Understandability	—	—	247	91.2	257	91.2	177	89.1	210	90.4	250	91.2	336	89.9	250	91.2
Internal Consistency	153	78.3	233	90.6	235	90.6	233	90.6	201	89.9	250	90.5	351	89.5	233	90.6
Efficiency & Size + Accuracy	187	90.2	229	91.6	229	91.6	229	91.6	187	90.2	250	91.4	362	90.7	229	91.6
US&UB + Accuracy	188	90.5	237	92.1	237	92.1	237	92.1	193	90.6	250	92.0	353	90.4	237	92.1
Efficiency & Size + US&UB	241	91.6	246	91.7	—	—	60	58.9	205	91.2	250	91.6	379	90.0	246	91.7
Efficiency & Size + Accuracy + US&UB	209	92.0	198	91.2	268	91.7	198	91.2	211	92.0	250	91.8	361	91.0	211	92.0

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	223	84.3	225	84.5	250	80.3	343	85.3	343	85.3
Efficiency	—	—	—	—	—	—	—	—	201	85.6	250	85.9	303	85.3	303	85.3
Accuracy	217	92.1	189	92.0	189	92.0	189	92.0	199	92.0	250	91.7	289	91.2	199	92.0
Understanding	177	90.6	201	91.8	217	91.9	201	91.8	233	92.0	250	91.9	332	91.0	217	91.9
Understandability	236	91.0	—	—	—	—	184	90.8	233	91.0	250	91.2	343	90.6	250	91.2
Internal Consistency	207	90.9	208	90.9	217	91.0	207	90.9	203	90.8	250	90.4	312	89.3	208	90.9
Efficiency & Size + Accuracy	181	91.8	171	91.7	179	91.8	171	91.7	190	91.9	250	91.6	303	91.1	181	91.8
US&UB + Accuracy	178	91.8	182	91.9	195	92.1	181	91.9	196	92.2	250	92.0	313	91.6	195	92.1
Efficiency & Size + US&UB	170	90.9	182	91.3	192	91.6	158	88.6	210	91.6	250	91.2	299	90.8	192	91.6
Efficiency & Size + Accuracy + US&UB	227	92.0	190	91.9	191	91.9	73	72.6	197	91.9	250	91.7	332	91.3	197	91.9

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.3: GRAEL-1 ATIS - 20 Agents - Crossover - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	139	90.4	140	90.3	142	90.5	140	90.3	102	83.1	200	90.5	—	—	140	90.3
Efficiency	139	90.8	136	90.9	—	—	112	90.9	102	89.9	200	90.6	—	—	139	90.8
Accuracy	139	91.3	84	91.3	84	91.3	84	91.3	102	91.1	200	91.0	—	—	102	91.1
Understanding	139	91.1	81	91.2	81	91.2	28	65.0	102	91.3	200	91.0	—	—	102	91.3
Understandability	139	90.1	119	90.3	120	90.3	97	87.7	102	90.6	200	90.6	—	—	120	90.3
Internal Consistency	139	89.3	—	—	—	—	166	90.1	102	90.4	200	90.6	—	—	200	90.6
Efficiency & Size + Accuracy	139	91.1	98	91.4	123	91.2	94	91.3	102	91.4	200	91.0	—	—	123	91.2
US&UB + Accuracy	139	91.0	99	91.2	107	91.3	99	91.2	102	91.4	200	91.0	—	—	107	91.3
Efficiency & Size + US&UB	139	90.9	105	91.2	110	91.2	100	91.3	102	91.4	200	91.0	—	—	110	91.2
Efficiency & Size + Accuracy + US&UB	139	90.8	100	91.0	101	91.0	100	91.0	102	91.1	200	91.0	—	—	102	91.1

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.4: GRAEL-1 ARTS - 10 Agents - Single Epoch - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	167	83.6	121	76.9	200	84.6	231	82.9	231	82.9
Efficiency	—	—	—	—	—	—	161	84.1	140	80.3	200	83.0	221	82.9	221	82.9
Accuracy	89	90.6	112	91.9	112	91.9	112	91.9	102	92.0	200	91.0	201	91.0	112	91.9
Understanding	116	82.3	155	91.9	155	91.9	155	91.9	112	82.1	200	91.4	219	91.0	155	91.9
Understandability	—	—	—	—	—	—	129	90.7	123	90.8	200	89.6	201	89.6	201	89.6
Internal Consistency	—	—	152	90.8	158	90.7	141	90.5	132	90.2	200	89.5	211	88.9	158	90.7
Efficiency & Size + Accuracy	96	82.3	133	91.6	133	91.6	133	91.6	134	91.6	200	91.0	211	91.1	133	91.6
US&UB + Accuracy	77	89.9	96	91.0	131	92.0	131	92.0	111	91.7	200	90.9	199	90.9	131	92.0
Efficiency & Size + US&UB	124	90.9	135	91.8	158	91.8	134	91.7	121	91.4	200	91.0	222	90.8	135	91.8
Efficiency & Size + Accuracy + US&UB	127	91.3	139	91.7	139	91.7	139	91.7	134	91.8	200	90.5	217	90.7	139	91.7

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.5: GRAEL-1 ATTS - 10 Agents - Splicing - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	153	82.3	131	85.3	200	84.9	215	84.8	215	84.8
Efficiency	—	—	—	—	163	85.3	135	85.1	119	84.9	200	85.0	201	85.1	200	85.0
Accuracy	89	92.1	103	92.2	103	92.2	103	92.2	95	92.2	200	—	195	90.7	103	92.2
Understanding	129	91.9	109	91.8	109	91.8	109	91.8	102	91.8	200	90.7	201	90.7	109	91.8
Understandability	143	90.7	—	—	—	—	136	90.2	141	90.6	200	90.0	209	89.5	200	90.0
Internal Consistency	173	90.5	166	90.2	—	—	119	89.9	138	90.4	200	90.0	209	90.0	173	90.5
Efficiency & Size + Accuracy	103	90.9	112	90.7	144	91.5	111	90.6	121	91.8	200	90.5	201	90.5	121	91.8
US&UB + Accuracy	59	78.5	100	92.1	100	92.1	100	92.1	99	92.2	200	—	198	90.9	100	92.1
Efficiency & Size + US&UB	134	92.0	139	91.9	139	91.9	105	90.5	119	91.8	200	—	199	90.7	139	91.9
Efficiency & Size + Accuracy + US&UB	107	89.7	126	91.7	163	91.8	126	91.7	131	91.5	200	90.8	209	90.7	131	91.5

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.6: GRAEL-1 ATIS - 10 Agents - Crossover - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	47	90.4	45	90.4	125	90.5	45	90.4	45	90.4	100	90.5	—	—	47	90.4
Efficiency	47	90.4	43	90.6	49	90.6	42	90.7	45	90.6	100	90.7	—	—	47	90.4
Accuracy	47	90.5	47	90.5	97	90.6	44	90.6	45	90.6	100	90.7	—	—	47	90.5
Understanding	47	90.5	43	90.8	49	90.7	47	90.5	45	90.6	100	90.7	—	—	47	90.5
Understandability	47	90.5	47	90.5	—	—	42	90.7	45	90.4	100	90.5	—	—	47	90.5
Internal Consistency	47	90.5	—	—	—	—	42	90.6	45	90.4	100	90.7	—	—	100	90.7
Efficiency & Size + Accuracy	47	90.4	56	90.7	—	—	43	90.6	45	90.4	100	90.5	—	—	56	90.7
US&UB + Accuracy	47	90.5	43	90.8	97	90.6	43	90.8	45	90.6	100	90.7	—	—	47	90.5
Efficiency & Size + US&UB	47	90.5	73	90.7	—	—	44	90.6	45	90.4	100	90.6	—	—	73	90.7
Efficiency & Size + Accuracy + US&UB	47	90.4	42	90.6	—	—	41	90.8	45	90.6	100	90.7	—	—	47	90.4

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.7: GRAEL-1 ARTIS - 5 Agents - Single Epoch - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	95	90.3	91	90.4	100	90.6	160	89.1	160	89.1
Efficiency	—	—	95	91.0	97	91.1	93	91.0	85	91.0	100	90.7	153	89.0	97	91.1
Accuracy	39	91.2	39	91.2	40	91.2	39	91.2	45	90.9	100	90.6	131	90.5	40	91.2
Understanding	62	91.1	62	91.1	81	90.9	62	91.1	61	91.1	100	90.8	134	90.4	62	91.1
Understandability	130	90.1	—	—	—	—	82	88.0	72	89.4	100	90.0	139	90.0	130	90.1
Internal Consistency	—	—	—	—	—	—	75	90.1	82	90.3	100	90.4	137	89.3	137	89.3
Efficiency & Size + Accuracy	48	82.3	87	91.0	95	90.8	87	91.0	73	91.2	100	91.1	129	90.3	87	91.0
US&UB + Accuracy	46	91.3	63	91.1	114	90.6	63	91.1	51	91.2	100	90.7	121	90.4	63	91.1
Efficiency & Size + US&UB	75	89.3	90	91.0	90	91.0	90	91.0	80	91.2	100	90.7	130	90.8	90	91.0
Efficiency & Size + Accuracy + US&UB	45	80.1	62	91.1	63	91.1	62	91.1	60	91.1	100	90.8	130	90.2	62	91.1

Legend		
	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.8: GRAEL-1 ATIS - 5 Agents - Splicing - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	—	—	71	86.3	100	88.1	120	87.3	120	87.3
Efficiency	—	—	—	—	—	—	—	—	76	90.0	100	90.3	118	90.2	118	90.2
Accuracy	48	91.2	60	91.1	60	91.1	60	91.1	51	91.1	100	91.2	105	91.2	60	91.1
Understanding	64	91.1	70	91.1	70	91.1	70	91.1	62	91.1	100	90.8	109	90.7	70	91.1
Understandability	68	90.1	63	90.3	119	89.1	60	90.5	58	90.2	100	90.4	110	89.6	68	90.1
Internal Consistency	70	90.3	—	—	—	—	78	90.6	63	89.6	100	88.3	113	90.0	100	88.3
Efficiency & Size + Accuracy	82	91.0	50	91.0	76	91.0	94	90.7	46	91.0	100	90.7	109	90.4	82	91.0
US&UB + Accuracy	103	91.1	94	90.9	94	90.9	84	91.1	52	91.1	100	91.1	105	90.9	94	90.9
Efficiency & Size + US&UB	57	91.2	64	91.2	68	91.2	64	91.2	58	91.1	100	90.7	101	90.7	64	91.2
Efficiency & Size + Accuracy + US&UB	68	91.2	55	91.2	55	91.2	55	91.2	49	91.1	100	90.6	107	90.5	55	91.2

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.9: GRAEL-1 ATIS - 5 Agents - Crossover - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	161	64.3	225	90.5	225	90.5	225	90.5	198	90.5	300	90.1	—	—	225	90.5
Efficiency	161	79.3	—	—	—	—	215	88.6	198	90.0	300	90.0	—	—	300	90.0
Accuracy	161	90.5	166	90.7	216	91.2	166	90.7	198	90.9	300	91.0	—	—	198	90.9
Understanding	161	90.0	196	91.0	197	91.0	163	90.2	198	91.0	300	90.6	—	—	197	91.0
Understandability	161	76.7	231	91.0	231	91.0	182	86.6	198	91.1	300	90.1	—	—	231	91.0
Internal Consistency	161	80.6	214	90.9	217	90.6	214	90.9	198	90.6	300	90.7	—	—	214	90.9
Efficiency & Size + Accuracy	161	89.0	213	90.0	—	—	213	90.0	198	90.4	300	90.6	—	—	213	90.0
US&UB + Accuracy	161	90.2	233	91.1	233	91.1	233	91.1	198	91.0	300	90.9	—	—	233	91.1
Efficiency & Size + US&UB	161	79.3	206	90.9	212	90.8	206	90.9	198	90.7	300	90.6	—	—	206	90.9
Efficiency & Size + Accuracy + US&UB	161	79.6	225	90.8	225	90.8	225	90.8	198	90.9	300	90.1	—	—	225	90.8

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.10: GRAEL-1 ATIS - 50 Agents - Single Epoch - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	332	88.9	310	87.8	350	86.8	431	85.8	431	85.8
Efficiency	—	—	—	—	—	—	312	89.3	308	89.2	350	89.3	422	89.5	422	89.5
Accuracy	217	84.3	278	92.2	279	92.2	190	79.8	253	92.1	350	91.9	401	91.2	278	92.2
Understanding	214	78.6	275	91.3	290	92.1	261	90.7	263	91.1	350	92.0	410	90.8	275	91.3
Understandability	263	91.5	280	90.7	—	—	280	90.7	271	91.2	350	90.3	417	90.2	280	90.7
Internal Consistency	—	—	—	—	—	—	150	52.7	302	88.9	350	89.3	413	88.9	413	88.9
Efficiency & Size + Accuracy	292	92.0	264	91.2	275	91.3	180	65.9	293	91.9	350	91.8	399	90.3	292	92.0
US&UB + Accuracy	231	86.8	257	91.8	266	92.1	149	62.7	271	92.2	350	92.0	409	91.8	266	92.1
Efficiency & Size + US&UB	207	70.2	282	91.7	302	91.9	130	58.7	299	92.0	350	91.9	423	91.1	299	92.0
Efficiency & Size + Accuracy + US&UB	247	89.4	271	92.0	277	92.0	228	81.4	261	91.6	350	91.6	412	90.8	271	92.0

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.11: GRAEL-1 ATIS - 50 Agents - Splicing - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	337	82.3	312	80.9	300	80.1	412	82.1	412	82.1
Efficiency	—	—	—	—	—	—	242	82.1	249	83.1	300	84.2	398	81.6	398	81.6
Accuracy	191	85.9	229	91.1	246	92.1	158	75.9	250	92.2	300	92.1	351	91.8	246	92.1
Understanding	116	64.3	231	91.9	231	91.9	182	89.4	213	92.0	300	91.8	361	90.7	231	91.9
Understandability	—	—	—	—	—	—	142	59.6	213	89.9	300	90.0	355	89.7	355	89.7
Internal Consistency	—	—	—	—	—	—	211	83.3	249	89.7	300	89.9	365	90.2	365	90.2
Efficiency & Size + Accuracy	192	91.2	186	90.8	216	91.6	96	60.9	209	91.8	300	91.3	349	90.6	209	91.8
US&UB + Accuracy	244	92.2	232	92.2	232	92.2	165	73.2	234	92.2	300	91.8	331	91.3	234	92.2
Efficiency & Size + US&UB	221	91.1	246	91.9	246	91.9	211	90.8	242	91.8	300	91.6	359	90.7	246	91.9
Efficiency & Size + Accuracy + US&UB	151	72.1	267	92.0	267	92.0	246	91.2	241	91.3	300	92.0	361	91.5	267	92.0

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.12: GRAEL-1 ATIS - 50 Agents - Crossover - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	195	88.3	266	90.5	—	—	223	90.6	241	90.9	300	90.6	—	—	266	90.5
Efficiency	195	88.1	227	90.9	227	90.9	227	90.9	241	90.8	300	90.7	—	—	227	90.9
Accuracy	195	90.9	242	91.1	243	91.1	242	91.1	241	91.1	300	91.2	—	—	242	91.1
Understanding	195	90.3	212	90.6	221	90.9	212	90.6	241	90.8	300	90.5	—	—	221	90.9
Understandability	195	76.9	—	—	—	—	247	89.1	241	88.9	300	90.3	—	—	300	90.3
Internal Consistency	195	79.8	—	—	—	—	205	82.1	241	83.9	300	90.6	—	—	300	90.6
Efficiency & Size + Accuracy	195	90.6	222	90.7	222	90.7	222	90.7	241	90.9	300	90.6	—	—	222	90.7
US&UB + Accuracy	195	90.9	240	91.0	240	91.0	240	91.0	241	91.0	300	90.7	—	—	240	91.0
Efficiency & Size + US&UB	195	83.3	254	90.7	—	—	254	90.7	241	90.4	300	90.8	—	—	254	90.7
Efficiency & Size + Accuracy + US&UB	195	88.8	240	91.0	240	91.0	208	90.8	241	91.1	300	90.8	—	—	240	91.0

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.13: GRAEL-1 ATIS - 100 Agents - Single Epoch - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	351	90.6	353	90.2	400	90.5	478	90.1	478	90.1
Efficiency	—	—	—	—	—	—	346	87.9	359	88.0	400	88.2	468	90.3	468	90.3
Accuracy	275	82.3	340	92.0	340	92.0	174	55.7	345	92.0	400	91.8	451	90.6	340	92.0
Understanding	246	78.1	310	90.5	331	91.9	310	90.5	322	91.8	400	90.8	448	90.6	322	91.8
Understandability	258	79.8	—	—	—	—	254	79.8	322	90.8	400	90.0	453	89.8	400	90.0
Internal Consistency	344	90.2	361	90.3	—	—	303	84.3	350	90.3	400	89.5	461	89.0	361	90.3
Efficiency & Size + Accuracy	166	59.3	288	90.3	291	90.5	165	59.5	310	91.0	400	91.0	434	89.9	291	90.5
US&UB + Accuracy	265	84.3	330	91.0	342	91.8	279	85.7	352	92.0	400	92.0	441	91.5	342	91.8
Efficiency & Size + US&UB	381	91.9	370	92.0	370	92.0	238	60.6	374	92.0	400	91.8	455	90.3	374	92.0
Efficiency & Size + Accuracy + US&UB	244	76.3	300	91.1	320	91.9	275	88.3	329	91.9	400	90.9	423	89.9	320	91.9

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.14: GRAEL-1 ATIS - 100 Agents - Splicing - Halting Points and F-scores on Test Set

Fitness Function	FS vs TS		FA vs TS		FA vs FS		Plateau Detect.		Under-Stand.		Limit LG		Limit Gen.		Majority Voting	
	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F	HP	F
Size	—	—	—	—	—	—	379	87.1	387	87.9	400	88.3	440	87.2	440	87.2
Efficiency	—	—	—	—	—	—	354	90.3	350	90.2	400	88.9	431	86.9	431	86.9
Accuracy	327	90.7	347	92.0	347	92.0	248	78.3	340	91.9	400	91.6	425	90.9	347	92.0
Understanding	271	84.1	329	90.6	340	91.6	304	89.3	274	81.9	400	91.6	419	91.5	329	90.6
Understandability	—	—	—	—	—	—	266	85.3	351	87.6	400	88.4	403	88.4	403	88.4
Internal Consistency	—	—	—	—	—	—	121	45.9	250	70.6	400	88.1	409	88.0	409	88.0
Efficiency & Size + Accuracy	260	75.2	310	90.8	333	92.0	257	78.6	340	92.1	400	90.9	419	90.7	333	92.0
US&UB + Accuracy	309	91.5	333	92.0	368	92.1	243	74.9	300	91.0	400	91.9	427	91.8	333	92.0
Efficiency & Size + US&UB	370	91.7	335	91.9	335	91.9	260	81.0	353	91.9	400	90.5	401	90.5	353	91.9
Efficiency & Size + Accuracy + US&UB	316	91.2	311	91.1	347	91.9	306	91.1	310	90.9	400	91.1	413	90.9	316	91.2

Legend	FS	Full Society Accuracy
	TS	Training Set Accuracy
	FA	Fittest Agent Accuracy
	HP	Halting Point
	F	$F_{\beta=1}$ -score of fittest agent on test set (%)
	US	Understanding
	UB	Understandability

Table D.15: GRAEL-1 ATIS - 100 Agents - Crossover - Halting Points and F-scores on Test Set



GRAEL-1 WSJ Full Results Tables

E.1 20 Agents

E.2 10 Agents

E.3 5 Agents

E.4 50 Agents

E.5 100 Agents

	US		US+AC	
	HP	$F_{\beta=1}$	HP	$F_{\beta=1}$
FS vs TS	—	—	252	80.5
FA vs TS	—	—	184	81.4
FA vs FS	—	—	184	81.4
Plateau	—	—	184	81.4
Understanding	153	81.2	149	81.3
Limit LG	250	81.2	250	80.5
Limit Gen	342	79.8	333	79.7
Majority Voting	—	—	184	81.4

FS Full Society Accuracy
TS Training Set Accuracy
FA Fittest Agent Accuracy
LG Language Games
HP Halting Point
US Understanding Fitness Function
AC Accuracy Fitness Function

Table E.1: GRAEL-1 WSJ - 20 Agents - Crossover - Halting Points and F-scores on Test Set

	US		US+AC	
	HP	$F_{\beta=1}$	HP	$F_{\beta=1}$
FS vs TS	—	—	103	80.9
FA vs TS	—	—	118	81.2
FA vs FS	—	—	—	—
Plateau	—	—	118	81.2
Understanding	122	81.1	113	81.1
Limit LG	200	80.5	200	—
Limit Gen	211	80.6	181	80.8
Majority Voting	—	—	118	81.2

FS Full Society Accuracy
TS Training Set Accuracy
FA Fittest Agent Accuracy
LG Language Games
HP Halting Point
US Understanding Fitness Function
AC Accuracy Fitness Function

Table E.2: GRAEL-1 WSJ - 10 Agents - Crossover - Halting Points and F-scores on Test Set

	US		US+AC	
	HP	$F_{\beta=1}$	HP	$F_{\beta=1}$
FS vs TS	—	—	—	—
FA vs TS	—	—	77	80.3
FA vs FS	—	—	—	—
Plateau	—	—	59	80.1
Understanding	72	80.0	79	80.3
Limit LG	100	80.1	100	80.4
Limit Gen	122	80.2	115	80.5
Majority Voting	—	—	100	80.4

FS Full Society Accuracy
TS Training Set Accuracy
FA Fittest Agent Accuracy
LG Language Games
HP Halting Point
US Understanding Fitness Function
AC Accuracy Fitness Function

Table E.3: GRAEL-1 WSJ - 5 Agents - Crossover - Halting Points and F-scores on Test Set

	US		US+AC	
	HP	$F_{\beta=1}$	HP	$F_{\beta=1}$
FS vs TS	—	—	223	81.4
FA vs TS	—	—	199	81.3
FA vs FS	—	—	—	—
Plateau	—	—	138	81.2
Understanding	175	81.3	214	81.4
Limit LG	250	81.3	250	81.4
Limit Gen	301	80.8	298	81.0
Majority Voting	—	—	223	81.4

FS Full Society Accuracy
TS Training Set Accuracy
FA Fittest Agent Accuracy
LG Language Games
HP Halting Point
US Understanding Fitness Function
AC Accuracy Fitness Function

Table E.4: GRAEL-1 WSJ - 50 Agents - Crossover - Halting Points and F-scores on Test Set

	US		US+AC	
	HP	$F_{\beta=1}$	HP	$F_{\beta=1}$
FS vs TS	—	—	260	81.4
FA vs TS	—	—	266	81.4
FA vs FS	—	—	260	81.4
Plateau	—	—	260	81.4
Understanding	255	81.4	251	81.4
Limit LG	300	81.4	300	81.3
Limit Gen	342	81.3	359	81.3
Majority Voting	—	—	260	81.4

FS Full Society Accuracy
TS Training Set Accuracy
FA Fittest Agent Accuracy
LG Language Games
HP Halting Point
US Understanding Fitness Function
AC Accuracy Fitness Function

Table E.5: GRAEL-1 WSJ - 100 Agents - Crossover - Halting Points and F-scores on Test Set



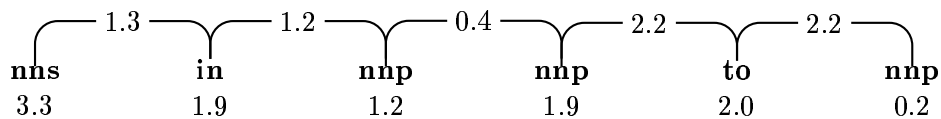
124 Unique Rules in GRAEL-2 Test Set

ADJP → JJ PP
ADJP → JJS
ADJP → RB RB PP
ADJP → RBS JJ
ADVP → RB PP
FRAG → ADVP NP
FRAG → NP INTJ
FRAG → NP NP NP NP
FRAG → NP NP PP PP
FRAG → NP PP ADVP
FRAG → PP PP VP
FRAG → PP VP
FRAG → WHNP PP
NP → CC NNP CC NNP
NP → CD
NP → CD CD CD RB
NP → CD JJ
NP → DT ADJP NN
NP → DT CD CD CD NN
NP → DT CD CD NN
NP → DT CD NN NN
NP → DT JJ NNP NN
NP → DT JJ X NNS
NP → DT JJS JJ NN NNS
NP → DT JJS NNS
NP → DT NN NN SYM
NP → DT NN SYM CD
NP → DT NN SYM SYM SYM CD CD
NP → DT NNP NNP CD CD NN
NP → DT NNP NNP NN
NP → DT NNP NNP NNP NN
NP → DT NNP NNP NNP NNS
NP → DT NX
NP → JJ JJ NN
NP → JJ NNP NNS
NP → JJS NN NN
NP → JJS NN NNS
NP → JJS NNS
NP → NN NN NN
NP → NN NN NNP CD CD CD CD
NP → NN NN SYM
NP → NN NNP NNP
NP → NN NNP NNP CD CD
NP → NN NNP NNP CD CD CD
NP → NN NNP CD CD
NP → NNP CD CD CD
NP → NNP CD CD CD CD
NP → NNP NNP NN
NP → NNP NNP NNS
NP → NNP NNS
NP → NNPS
NP → NP CC ADVP NP
NP → NP CC PP PP
NP → NP NNS
NP → NP NP PP PP PP
NP → NP NP PP X PP
NP → NP PP ADJP
NP → NP PP NP
NP → NP PP PP ADVP NP
NP → NP PP PP NP ADVP
NP → NP PP PP PP NP
NP → NP PP PP PP VP
NP → NP PP VP VP
NP → NP SBAR PP
NP → QP NNS QP
NP → RB DT NNP NNP NNS
NP → RB VBG NNS
NP → WDT NNS
NX → NN
NX → NN NN
NX → NX CC NX
PP → ADVP IN NP
PP → IN
PP → IN ADJP
PP → IN CC IN NP
PP → NP IN S
PP → PP CC PP
PP → PP PP
PP → TO ADVP
PP → TO INTJ
PRT → RP
QP → CC JJR
QP → CD CD CD CD
QP → RB JJR IN CD CD CD
S → FRAG
S → NP ADVP VP
S → S S
SBAR → WHNP SQ
SBAR → XXX
SBARQ → PP WHNP SQ
SQ → INTJ MD NP VP
SQ → VBP NP NP PP PP NP SBAR
SQ → VBP NP PP
SQ → VBZ NP PP
SQ → X VBZ NP ADJP
VP → NN NP
VP → VB
VP → VB ADVP NP
VP → VB ADVP PP ADJP PP
VP → VB NP ADVP
VP → VB NP PP ADVP
VP → VB NP PP PP
VP → VB NP PP X PP
VP → VB PP PP NP
VP → VB PRT NP
VP → VBD NP
VP → VBN NP
VP → VBP FRAG
VP → VBP PP PP NP
VP → VBP RB VP
VP → VBP VP
VP → VBZ VP
VP → XXX
WHNP → NP PP PP PP
WHNP → WHNP ADJP
WHNP → WHNP PP PP NP SBAR
WHNP → WHNP PP PP PP PP
WHNP → XXX
X → JJ
X → S
X → SBARQ
X → VBZ NP
X → WRB IN

G

Grammar Induction failure

We try to induce a tree-structure for a fragmentary sentence like *flights from Los Angel's to Toronto*. The original tree-structure for this sentence is displayed in FigureG.1.



Starting off with this situation, there are two bigrams with the same lexical attraction value. We choose the one with the largest amount of information content and create a constituent:

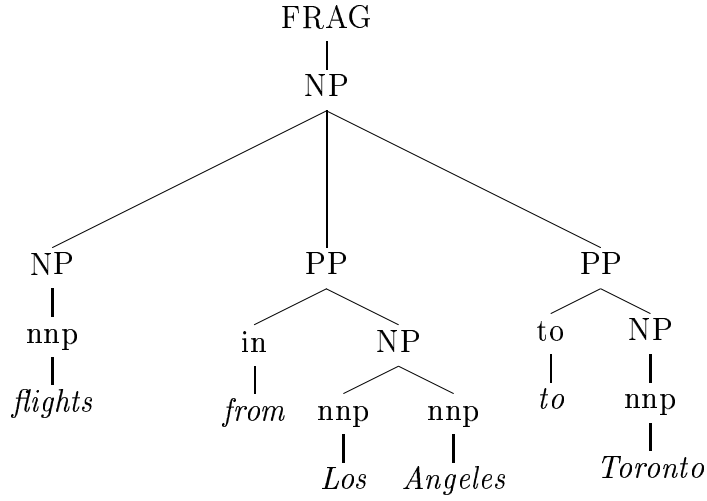
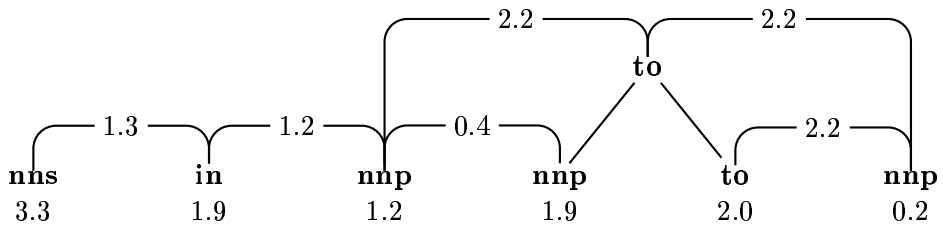
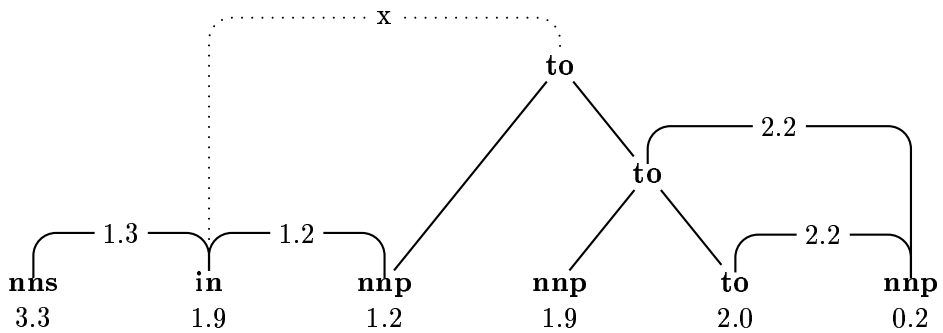


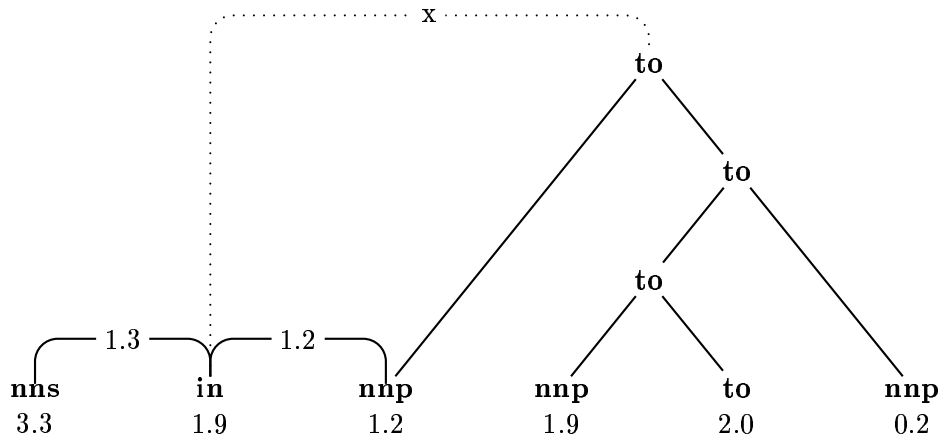
Figure G.1: Original Tree-Structure



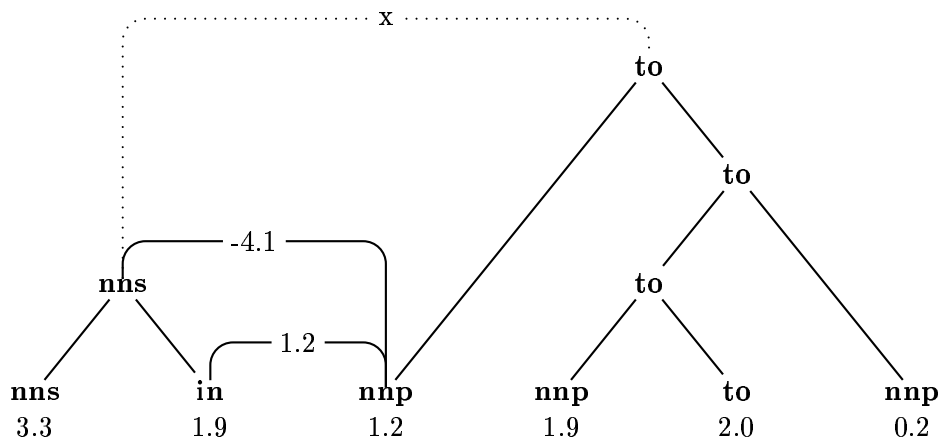
The next sequence under consideration is “**nnp to**”, which creates a new leaf to the constituent headed by **to**:



Next we choose the highest attachment point for the sequence “*to nnp*”. This provides a conflict where a node is headed by two other nodes. Resolving this situation, we obtain the following structure:



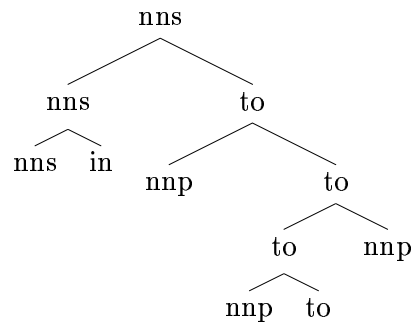
The “*nns in*” sequence is next:



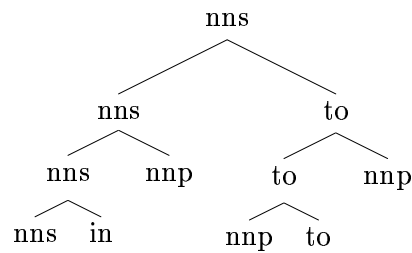
This leaves us with only one linking to resolve and a troublesome one at that, since it presents us with the added difficulty of two leaf nodes to join in one construct. The grammar induction tries to minimally interfere with the segmentation

and labeling we committed to earlier. We can identify three options for joining these two structures:

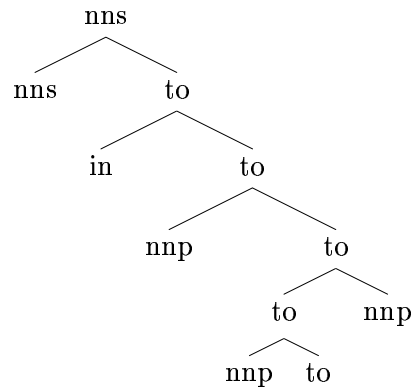
The first option just introduces a new top-level-item and joins whatever constituents are left.



The second option attaches the **nns**-constituent as a leaf node on the **to**-constituent. This however changes the label of the root-node which is not allowed.



The 3rd option attaches the **to**-constituent as a leaf-node of the **nns**-constituent. This is allowed, since it does not change the root-node label. Since the first option is only used when neither option is valid, the grammar induction uses this constituent.



The structure that is induced is however totally wrong: Figure G.2 shows the tree-structure with the words attached. The flattened tree (Figure G.3) and re-labeled flattened tree (Figure G.4) appear a bit better, but overall, the grammar induction fails on this sentence.

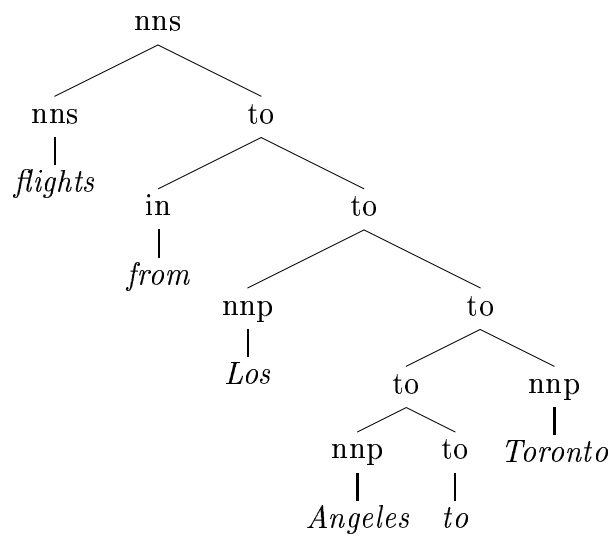


Figure G.2: Binary Branching Output

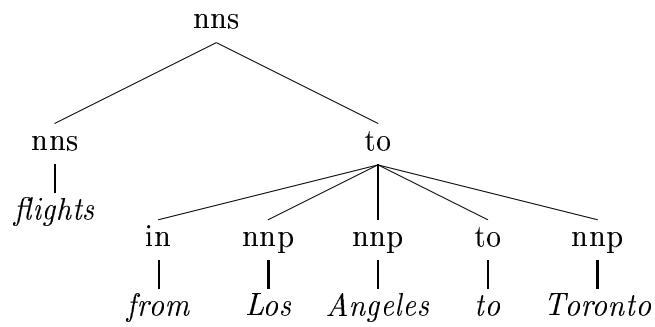


Figure G.3: Flattened Tree

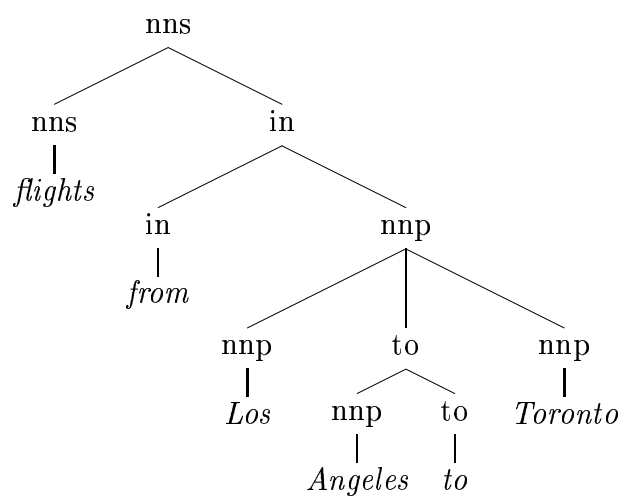


Figure G.4: Re-labeled flattened tree

H

Symbols used in the Penn Treebank

ADJP	Adjectival Phrase	ADVP	Adverbial Phrase
CONJP	Conjunction Phrase	FRAG	Fragmentary Phrase
LST	List marker	NP	Noun Phrase
NP-SBJ	Subject-NP	NX	Head of NP in complex NP
PP	Prepositional Phrase	PRN	Parenthetical
PRT	Particle	QP	quantitative Phrase
RRC	Reduced Relative Clause	SBAR	Conjoined S-Phrase
SBARQ	WH-Question	SINV	Inverted decl. sentence
SQ	Inverted yes/no question	VP	Verb Phrase
WHADVP	WH-Adverbial Phrase	WHNP	WH-Noun Phrase
WHPP	Wh-prepositional Phrase	X	Unknown

Table H.1: Syntactic Category Labels in the Penn Treebank

Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential	there there is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	both the boys
POS	possessive ending	friend's
PRP	personal pronoun	I, he, it
PRP\$	possessive pronoun	my, his
RB	adverb	usually
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give up
TO	to	to go, to him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP\$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when
from http://www.mozart-oz.org/mogul/doc/lager/brill-tagger/penn.html		

Table H.2: The Penn Treebank Tag Set



Set of 100 fixed Meanings

I.1 4 Attributes

happy(Agent0) sad(Agent1)
stinky(dog) hungry(tiger)

I.2 96 Relations

I.2.1 18 x [+ animate] [+ animate]

and(fish,Agent3) not(dog,Agent2)
and(Agent2,Agent9) not(Agent3,Agent4)
eat(bear,dog) fight(tiger,Agent5)
hit(ape,tiger) see(horse,snake)
runto(dog,pig) catch(Agent0,ape)
kick(Agent2,Agent1) offer(Agent3,fish)

```
fear(Agent4,lion)    chase(Agent5,tiger)
throw(Agent6,duck)  talk(Agent7,Agent2)
know(Agent8,Agent1) love(Agent9,Agent0)
```

I.2.2 18 x [+ animate] [- animate]

```
and(Agent4,stick)    catch(pig,leaf)
chase(Agent6,flower) eat(horse,nut)
fear(lion,bone)      fight(Agent1,society)
hit(Agent9,bed)      kick(duck,nut)
know(Agent9,bone)    love(Agent4,lake)
love(fish,river)    not(ape,sand)
not(Agent0,bed)      offer(horse,leaf)
runto(Agent0,bone)   see(horse,coat)
talk(Agent7,society) throw(ape,bone)
```

I.2.3 60 Complex Relationships

```
hit(happy(agent9),egg)
kick(sad(snake),duck)
know(happy(bear),agent5)
love(hungry(agent9),egg)
runto(sad(snake),bone)
talk(hungry(agent3),ape)
catch(chase(agent2,ape),duck)
chase(fish,love(agent9,horse))
eat(duck,know(tiger,agent2))
fear(stinky(agent4),happy(agent2))
fear(talk(agent9,agent1),fire)
fight(snake,know(agent2,lion))
fight(throw(agent2,fish),sand)
hit(eat(agent9,tree),egg)
know(agent8,(runto(agent9,river)))
know(agent8,hit(agent3,flower))
know(hungry(tiger),happy(bear))
```



```
love(agent8, eat(snake, nut))
love(hit(tiger, tree), ape)
not(dog, love(tiger, agent5))
not(talk(agent5, bear), bear)
runto(and(duck, fish), agent7)
runto(and(lion, agent9), house)
runto(bear, love(agent4, fruit))
runto(dog, chase(agent4, lion))
runto(horse, love(tiger, agent2))
runto(not(agent3, agent7), ape)
see(agent0, fight(agent1, fire))
see(ape, love(agent0, agent2))
see(snake, runto(agent1, dog))
talk(agent0, and(agent9, dog))
talk(agent9, see(agent2, fish))
throw(agent2, and(horse, pig))
throw(fish, hit(agent7, tree))
throw(snake, catch(agent4, fire))
and(hungry(agent5), love(agent8, pig))
catch(fight(stinky(agent5), bear), snake)
fight(happy(agent3), chase(agent1, tiger))
fight(sad(bear), fear(agent7, lion))
fight(stinky(dog), runto(agent8, sand))
know(horse, eat(sad(duck), seed))
know(stinky(bear), hit(agent8, agent5))
love(agent3, chase(happy(agent9), duck))
not(sad(agent6), and(bear, agent6))
runto(agent4, talk(agent4, sad(horse)))
runto(happy(bear), chase(agent1, agent8))
talk(agent9, runto(bear, stinky(horse)))
talk(stinky(agent2), love(bear, lake))
catch(sad(horse), see(sad(agent4), agent5))
chase(love(agent5, ape), chase(agent6, lion))
chase(see(know(ape, bed), agent9), pig)
chase(and(agent6, agent7), eat(tiger, pig))
kick(horse, chase(sad(fish), stinky(bear)))
love(stinky(ape), catch(happy(ape), seed))
see(and(bear, snake), hit(agent0, horse))
```

```
talk(catch(dog,catch(fish,house)),fire)
know(agent5(chase(agent0,
                love(agent4,
                    fight(duck,happy(pig))))))
hit(fish,talk(agent4,
              see(hungry(agent8),
                  kick(sad(agent4),snake))))
runto(sad(agent4),and(agent2,
                     and(and(agent3,tiger),
                          catch(agent4,stick))))
chase(offer(runto(eat(chase(stinky(agent6),
                          love(agent1,happy(duck))),
                    agent1),pig),flower),
      not(see(chase(fish,fruit),seed),tree),society)
```

J

Abbreviations

J.1 GRAEL Instantiations

GRAEL-1	Grammar Optimizations
GRAEL-2	Grammar Rule Discovery
GRAEL-3A-1	Unsupervised Grammar Induction. Pre-processing Stage by GIM. Parsing by PMPG
GRAEL-3B-1	Unsupervised Grammar Induction. No Pre-processing Stage. Parsing by PMPG
GRAEL-3AB-2	Unsupervised Grammar Induction. Parsing by GIM
GRAEL-4	Computational simulation of the emergence of grammar

J.2 Terms

OCB	Zero-Crossing Brackets	LP	Labeled Precision
A	Adding Nodes	LR	Labeled Recall
AA	Average Agent	M?	Mutation Type
AC	Accuracy on Validation Set	MBL	Memory-Based Learning
ADC	Adding, Deleting, Changing Nodes	MDL	Minimum-Description Length
ADC	Adding, Deleting, Changing Nodes (weighted)	NCM	Noisy Channel Mutation
ATIS	Air Travel Information System	P&P	Principles and Parameters
C	Changing Node Labels	PCFG	Probabilistic Context-Free Grammar
CFG	Context-Free Grammar	PMPG	Pattern-Matching Probabilistic Grammar
D	Deleting Nodes	RC	Random Crossover
DOP	Data-Oriented Parsing	SE	Single Epoch
E	E-language	SELFIT	Fitness function that selects candidate for evaluation
EF	Efficiency	SELPRO	Fitness function that selects candidates for reproduction
F-score	Weighted Average of LP and LR	SI	Size
FA	Fittest Agent	Sp	Splicing
FS	Full Society	TS	Training Set
GIM	Grammar Induction Method	UB	Understandability
GRAEL	Grammar Evolution	UG	Universal Grammar
I	I-language	US	Understanding (Accuracy on other agents)
IC	Internal Consistency	WSJ	Wall-Street Journal
IM	Internal Mutation	XO	Crossover
LAD	Language Acquisition Device	xv	Cross-Validation
LG	Language Game		
LGR	Language Game run		