

Bilexical Dependencies as an Intermedium for Data-Driven and HPSG-Based Parsing

Doctoral Dissertation by

Angelina Ivanova



Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo

Submitted for the degree of Philosophiae Doctor

November, 2015

© **Angelina Ivanova, 2015**

*Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 1690*

ISSN 1501-7710

All rights reserved. No part of this publication may be
reproduced or transmitted, in any form or by any means, without permission.

Cover: Hanne Baadsgaard Utigard.
Print production: John Grieg AS, Bergen.

Produced in co-operation with Akademika Publishing.
The thesis is produced by Akademika Publishing merely in connection with the
thesis defence. Kindly direct all inquiries regarding the thesis to the copyright
holder or the unit which grants the doctorate.

Abstract

Bilexical dependencies capturing asymmetrical lexical relations between heads and dependents are viewed as a practical representation of syntax that is well-suited for computation and intelligible for human readers. In the present work we use dependency representations as a bridge between data-driven and grammar-based parsing, both for cross-framework parser comparison and for parser integration.

We observe that the state of the art in dependency parsing for English is characterized by broad diversity of dependency representations and seek to systematize properties of various dependency formats pointing out their similarities and differences by carrying out qualitative and quantitative structural analysis and furthermore exploring learnability of four of these representations in automatic syntactic analysis. In addition to comparing syntactic dependencies along several evaluation measures for parsing, we also evaluate the representations in application to the negation resolution task.

Using a dependency representation extracted from HPSG structures we contrast three different approaches to parsing—data-driven dependency, phrase structure and a hybrid grammar-based—observe what trade-offs apply along accuracy, efficiency, coverage, and resilience to domain variation and show that explicit, hand-engineered grammatical knowledge helps in both accuracy and cross-domain parsing performance. We complement intrinsic parser evaluation with extrinsic comparison on the negation resolution and semantic dependency parsing tasks discovering that accuracy gains sometimes but not always translate into improved end-to-end performance.

A combination of complementary approaches is often a good strategy for achieving improvement. We explore parser integration as a method for advancing the efficiency of a grammar-based parser. Bilexical dependencies serve as an interface for enforcing constraints drawn from the output of the statistical, data-driven systems on the unification-based processing of the grammar-based parser. We experiment with confidence thresholding, filtering and parser ensembles for tackling the problem of selecting high-quality dependencies and propose a technique of static analysis as preliminary evaluation in navigating a large space of various combination setups. We choose configurations optimizing for speed, coverage and balancing the two metrics and carefully evaluate the trade-offs along efficiency, coverage, accuracy and domain-resilience.

Acknowledgements

I would like to express my deep gratitude to my advisors, Lilja Øvrelid and Stephan Oepen, for their encouragement, advice and patience. It is a great pleasure to work with them, I appreciate their warm and caring attitude and admire their insight into the field, professionalism and knowledge. Thank you for numerous discussions, lots of proofreading, continuous guidance and making time for meetings and answering my emails even during the busiest times. I would like to express my gratitude to the members of my reading committee, Joakim Nivre, Jennifer Foster and Martin Giese, for careful reading of the present thesis.

A very special thank to Rebecca Dridan for being always helpful with all sorts of questions, for guidance in the universe of DELPH-IN tools, positive attitude, encouragement and proofreading. Thanks for bringing up cheerfulness and humor in the daily office life, and reminding the group about opportunities of having a coffee or lunch in the sunshine whenever the sun was out.

I would like to warmly thank the former and present members of the Language Technology Group at the Department of Informatics at the University of Oslo, Jan Tore Lønning, Pierre Lison, Elisabeth Lien, Gordana Ilić Holen, Emanuele Lapponi, Arne Skjærholt, Milen Kouylekov, Gisle Ytrestøl, Erik Velldal, Jonathon Read, Murhaf Fares, Herman Ruge Jervell, Asbjørn Brændeland, Aleksander Øhrn, for a unique friendly and motivating environment. Thank you, Pierre, Arne, Eman, Milen, Gisle, for discussions on parsing and statistics and your help with all sorts of questions. Thank you, Elisabeth and Gordana, for your care and support.

I would like to acknowledge the support of the Norwegian Research Council. I am grateful to the Scientific Computing staff at the University of Oslo and the Norwegian Metacenter for Computational Science. Experimentation was made possible through access to the TITAN and ABEL high-performance computing facilities and the NorStore storage facilities at the University of Oslo.

I am very grateful to Gertjan van Noord for his guidance of my exchange project, invaluable feedback and critical comments. I would like to thank former and present members of Alfabinformatica at the University of Groningen, John Nerbonne, Gosse Bouma, Johan Bos, Simon Šuster, Valerio Basile, Kilian Evang, Noortje Venhuizen, Harm Brouwer, Daniël de Kok, Malvina Nissim, Çağrı Çöltekin, Leonie Bosveld, Gideon Kotzé, for making my stay in Groningen so memorable.

I owe a lot of thanks to Joakim Nivre for organizing my visit to his group at the Department of Linguistics and Philology at the Uppsala University on a very short notice. Special thanks to Jörg Tiedemann, Christian Hardmeier and Sara Stymne.

I also want to give a special thanks to Marco Kuhlmann, Bernd Bohnet, André Martins, Ryan McDonald, Richard Eckart de Castilho and Pedro Santos for helpful suggestions, and

advice related to parsing software.

I would like to thank participants of DELPH-IN community for motivation and inspiring discussions. I would particularly like to thank Dan Flickinger, Emily Bender, Yi Zhang, Rui Wang, Francis Bond and Ann Copestake for raising challenging questions and giving useful suggestions that helped to shape this thesis. Thanks to Petya Osenova, Antske Fokkens, Michael Goodman, Tim Baldwin, Tania Avgustinova, Montserrat Marimon, António Branco, Kiril Simov, Sanghoun Song, Lars Hellan, Zina Pozen, João Silva, Guy Emerson, Li Ling Tan and Ned Letcher for interaction and talks at DELPH-IN meetings and other conferences.

I would like to thank Tatiana, Maria, Thu, Le, Joachim, Jelke, Tina, Nathan, Amir, Bushra, Evgeny for their friendship and support. Thank you, Dima, for inspirational discussions about Python programming, statistics, natural language processing and sports. Thank you, Ke, for helping me to work out the details of a smoothing algorithm. I owe a lot of thanks to all my friends.

A very special thanks are dedicated to my parents for their love, help and eternal patience. I am very grateful to my in-law family for their kindness and the good times. And finally I would like to thank my dear husband for endless love, patience and positive outlook. You make me happy!

Contents

Abstract	iii
Acknowledgements	v
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Research questions	3
1.2 Contributions	4
1.3 Thesis outline	6
1.4 Publications	8
2 Background	9
2.1 Syntactic theories	9
2.1.1 Phrase-structure grammar	9
2.1.2 Dependency grammar	14
2.1.3 Head-Driven Phrase Structure Grammar	18
2.2 Syntactic parsing	23
2.2.1 Data-driven parsing	24
2.2.2 Deep grammar parsing	30
2.3 Parser combination	39
2.4 Parser evaluation measures	43
2.5 Linguistic resources	47
2.6 Summary	49
3 Syntactico-semantic dependencies	51
3.1 Why bilexical dependencies?	51
3.2 Motivation	52
3.3 Related work	53
3.4 Conversion procedure	55
3.4.1 Syntax: derivations to dependencies	56
3.4.2 Semantics: logical form to dependencies	58
3.4.3 Tokenization styles	63
3.4.4 Output format	64

3.5	Contrasting analysis	65
3.5.1	Variation in dependency representations	65
3.5.2	Qualitative analysis	70
3.5.3	Quantitative analysis	78
3.6	Summary	82
4	Contrasting parsing experiments	85
4.1	Related work	85
4.2	Experimental setup	90
4.3	Results	96
4.4	Error analysis	105
4.5	Extrinsic evaluation	109
4.6	Summary	112
5	Cross-framework parser evaluation	113
5.1	Motivation	113
5.2	Related work	114
5.3	Experimental setup	116
5.4	In-domain parsing results	121
5.5	Cross-domain parsing results	122
5.6	Error analysis	123
5.7	Sanity experiments	132
5.8	Extrinsic evaluation on negation resolution task	134
5.9	Extrinsic evaluation on semantic dependency parsing	137
5.10	Reflections on the SDP 2014 results	139
5.11	Summary	140
6	Parser combination	143
6.1	Introduction	143
6.2	Related work	144
6.2.1	Efficiency	144
6.2.2	Coverage	145
6.2.3	Accuracy	146
6.2.4	Parser combination for improved efficiency and accuracy	147
6.3	Hypothesis testing	148
6.4	Experimental setup	150
6.5	Tuning	155
6.5.1	Filtering parameters	157
6.5.2	Confidence thresholding	160
6.5.3	Ensembles of parsers	167
6.6	In-domain parser integration experiments	171
6.7	Cross-domain parser integration experiments	174
6.8	Summary	179

7 Conclusion 181

- 7.1 Main conclusions 181
- 7.2 Research questions revisited 184
- 7.3 Future research 185

Bibliography 189

List of Figures

2.1	Two syntactic analyses for the sentence “He saw a star with a telescope”	12
2.2	Constituency and dependency analysis of the sentence “Mike plays football”	15
2.3	Levels of sentence representation and modules of the MTT (Kahane, 2003)	16
2.4	Projective dependency tree example from Nivre (2008)	17
2.5	Non-projective dependency tree	17
2.6	The type hierarchy from Sag et al. (2003)	19
2.7	Syntactic analysis of the sentence “Salmon is a nutritious food” with the arc-eager algorithm	26
2.8	Lexical entry “ahead of” in the ERG lexicon	31
2.9	Derivation tree for the sentence “Every child probably has a funny toy”.	33
2.10	MRS for the sentence “Every child probably has a funny toy”	35
2.11	MRS for the sentence “Every child probably has a funny toy” in feature structure representation	36
2.12	EDS for the sentence “Every child probably has a funny toy”	37
2.13	Example of flat and extended analyses of the noun phrase “Air Force contract” from Vadas and Curran (2007)	38
2.14	Packed forest with sub-node decompositions	38
2.15	Crossing brackets	43
3.1	Simplified lexical entry from Alpino grammar for a finite transitive verb taking a direct object (Malouf and van Noord, 2004)	54
3.2	Dependency representations in DELPH-IN formats	56
3.3	ERG syntactic derivation tree for the sentence “A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.”	57
3.4	ERG Elementary Dependency Structure	59
3.5	Excerpt from the ERG configuration file	60
3.6	A WSJ sentence annotation using PTB-tokenization style	64
3.7	Dependency representations in CoNLL, Stanford, Enju, Prague, and DELPH-IN formats	66
5.1	Ambiguous token lattice	118
5.2	Character ranges for the phrase “in the ’70” which is tokenized differently in parser output with respect to the gold standard	119
5.3	Domain: WSJ. Distribution of dependency labels; precision of attachment of the most frequent dependency types	124

5.4	Domain: CB . Distribution of dependency labels; precision of attachment of the most frequent dependency types	124
5.5	Domain: SC . Distribution and precision of dependency; attachment of the most frequent dependency labels	124
5.6	Domain: VM . Distribution of dependency labels; precision of attachment of the most frequent dependency types	125
5.7	Domain: WS . Distribution of dependency labels; precision of attachment of the most frequent dependency types	125
5.8	Domain: WSJ . Dependency label precision relative to predicted dependency length; label recall relative to gold dependency length	126
5.9	Domain: CB . Dependency label precision relative to predicted dependency length; label recall relative to gold dependency length	126
5.10	Domain: SC . Dependency label precision relative to predicted dependency length; label recall relative to gold dependency length	127
5.11	Domain: VM . Dependency label precision relative to predicted dependency length; label recall relative to gold dependency length	127
5.12	Domain: WS . Dependency label precision relative to predicted dependency length; label recall relative to gold dependency length	127
5.13	Domain: WSJ . Distribution of PoS tags; accuracy of parsers for different PoS tags	129
5.14	Domain: CB . Distribution of PoS tags; accuracy of parsers for different PoS tags	129
5.15	Domain: SC . Distribution of PoS tags; accuracy of parsers for different PoS tags	129
5.16	Domain: VM . Distribution of PoS tags; accuracy of parsers for different PoS tags	130
5.17	Domain: WS . Distribution of PoS tags; accuracy of parsers for different PoS tags	130
6.1	Precision versus annotation rate for B&N on the development set	165
6.2	Tuning standard deviation and K of the KD-Fix algorithm on the development set	166

List of Tables

2.1	Types of variables in MRS	34
3.1	ERG configuration file that describes how many child nodes each grammar construction has and which daughter is the head	58
3.2	Tokenization pipeline during parsing with PET (Dridan, 2013b)	64
3.4	Summary of dependency formats	71
3.9	Statistics of the DM, EP and PT dependency graphs	82
4.8	Evaluation of automatic übertagging on the test set before conversion to PTB tokenization	95
4.9	Performance of the English Stanford Tagger and TnT taggers on the development set	95
4.10	Results of the automatic PTB PoS tagging and supertagging on the test set	95
4.11	Parsing results of Malt on SB, CD, DT and PA w/o scoring punctuation	96
4.12	Parsing results of MST on SB, CD, DT and PA w/o scoring punctuation	97
4.13	Parsing results of B&N on SB, CD, DT and PA w/o scoring punctuation	98
4.14	Parsing results of Malt on SB, CD, DT and PA including punctuation in the scoring	99
4.15	Parsing results of MST on SB, CD, DT and PA including punctuation in the scoring	100
4.16	Parsing results of B&N on SB, CD, DT and PA including punctuation in the scoring	101
4.17	Performance of Malt on the data annotated with PET-predicted supertags	102
4.18	Distribution of accuracy over PTB PoS tags when parsing with the Malt parser on the four dependency formats	105
4.19	Precision and recall of the labeling and attachment of the outgoing arcs for the coordinating conjunction for the Malt parser on different dependency formats with PTB PoS tags	106
4.20	Recall and precision of dependency relation and attachment for root when parsing with the Malt parser on the four dependency formats with PTB PoS tags	108
4.21	Conan Doyle corpus statistics (Morante and Blanco, 2012)	110
4.22	Performance of the negation resolution system UiO ₂ (Lapponi et al., 2012b) using dependency features from the analyses of B&N with CD, SB and DT dependencies in contrast with previous work	111
5.1	Sentence, token, and type counts for the DeepBank and Redwoods data sets	117

5.2	Parsing accuracy of B&N with three dependency schemes	117
5.3	Tuning of B&N on section 20 of DeepBank	120
5.4	Tagging accuracy, PARSEVAL F_1 , and dependency accuracy for the Berkeley parser on the development data	121
5.5	In-domain parsing experiments	121
5.6	Cross-domain parsing experiment	123
5.9	Number of total and correctly analyzed with ERG_a , Berkeley and B&N prepositional complements for various lexical types of the head—adjunct, noun and verb—on the WSJ domain	132
5.10	Number of total and correct analyses of prepositional complement structures	132
5.11	Coverage and dependency accuracies with PTB tokenization and either detailed or coarse lexical categories	133
5.12	Dependency accuracies computed with the eval.pl software for PTB tokenization and coarse lexical categories	134
5.13	Parse failures and token mismatches (‘gaps’), and tagging and dependency accuracy on the sub-set of the Conan Doyle development data	135
5.14	PET coverage on Conan Doyle training, development and test sets	135
5.15	Performance of the negation resolution system UiO ₂ on the development set using dependency features from the analyses of ERG_a , ERG_e and B&N with DT dependencies in contrast with the analyses of the Malt parser with SB dependencies from Lapponi et al. (2012b)	136
5.16	Performance of the negation resolution system UiO ₂ on the test set using dependency features from the analyses of ERG_a , ERG_e and B&N with DT dependencies in contrast with the analyses of the Malt parser with SB dependencies from Lapponi et al. (2012b)	136
5.17	SemEval 2014 open track results of the Priberam system on DM	138
5.18	SemEval 2014 open track results of the Priberam system on PAS	138
5.19	SemEval 2014 open track results of the Priberam system on PCEDT	139
5.20	Labeled Dependencies Including TOP Nodes	140
5.21	Unlabeled Dependencies Including TOP Nodes	141
6.1	Sentence and token counts and average sentence length for data sets of DeepBank 1.1 and Redwoods prepared with ERG 1214	152
6.2	Tokenization pipeline during parsing with PET (Dridan, 2013b)	154
6.3	Baseline and upper bound for parser combination on the development set	155
6.4	Evaluation of individual parsers on the development set	156
6.5	Results of parser combination without filtering on the development set	157
6.6	Tuning filtering parameter length of dependency	158
6.7	Filtering parameter part-of-speech tag of dependent	159
6.8	Filtering dependencies that have frequency less than 300 and precision less than 70%	160
6.9	Selecting only dependency types for which precision is minimum 80%	161
6.10	Selecting only dependencies of 11 types	161
6.11	Selecting only dependencies of 10 types	161

6.12	Selecting only dependencies of 8 types	162
6.13	Selecting only dependencies of 7 types	162
6.14	Parser integration: Turbo and PET	163
6.15	Correlation of per-dependency confidence scores of the B&N parser with correctness of attachment and labeling	164
6.16	Confidence thresholding with B&N	165
6.17	Correlation of MST per-dependency confidence scores with correctness of attachment and labeling	166
6.18	Confidence thresholding with MST	167
6.19	Ensemble BMT, tuning	168
6.20	Ensemble MMT, tuning	169
6.21	Ensemble BMMT, tuning	170
6.22	Evaluation of individual parsers on the test set	171
6.23	In-domain parser combination experiments	172
6.24	Evaluation of individual parsers on CB, VM, SC and WS	174
6.25	Out-of-domain parser combination experiments on CB	175
6.26	Out-of-domain parser combination experiments on VM	176
6.27	Out-of-domain parser combination experiments on SC	177
6.28	Out-of-domain parser combination experiments on WS	178

Chapter 1

Introduction

Isn't the language that ordinary people speak a wonderful thing that deserves to be studied in its own right?

— Martin Kay. *A Life in Language*

In the present work we focus on automatic syntactic analysis of natural language, called parsing, which assigns grammatical structures to an utterance. The main hypothesis motivating the parsing task is that grammatical structure contributes to meaning. There is a historic distinction between data-driven and grammar-based parsing approaches where the data-driven methods learn from labeled data relying on statistical algorithms and the grammar-based methods make parsing decisions with respect to an explicit, often hand-engineered system of syntactic knowledge usually with the aid of statistical means. The main hypothesis of data-driven parsing concerns the possibility for generalization about language using empirical data as a source of knowledge, and the most commonly exploited techniques employ machine learning algorithms that induce language models or grammars from annotated corpora. Grammar-based systems usually incorporate a hand-crafted grammar grounded in a linguistic theory and exploit statistical methods in a disambiguation module that selects the first-ranked analysis.

In the early days there was a strong tradition of grammar formalization in the field of computational linguistics that flourished after the publication of the monographs of Noam Chomsky in the tradition of Transformational Grammar (Chomsky, 1957, 1965, 1995, *inter alios*). Present-day Combinatory Categorical Grammar (CCG) (Steedman, 2000), Tree Adjoining Grammar (TAG) (Joshi et al., 1975), Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982) and Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) represent broadly used, theoretically well-motivated grammar formalisms originating from research initiatives from the period of 1950-1990s. A paradigm shift within the field of syntactic parsing began with automatic grammar induction as advances in technology and algorithms made it possible to acquire grammatical rules from manually annotated data that could still be extracted and comprehended by an interested user. Thus the Charniak-Johnson (Charniak and Johnson, 2005), Collins (Collins, 1999) and Berkeley (Petrov et al., 2006) parsers still make use of actual grammar rules, albeit often counting in the tens of thousands, based on principles of context-free grammar and training examples from a treebank. The next transition in the field has been to what we may call grammar-less parsing where explicit linguistic rules are not reconstructed during the training phase, with such prominent examples of purely data-driven parsers

as Malt (Nivre et al., 2007b) and MST (McDonald et al., 2005b). Following the predominance of research on statistical algorithms, the traditional grammar-oriented systems were modernized and augmented with various mechanisms for improved performance which currently makes the distinction between data-driven and grammar-based approaches less clear-cut today. At the moment there is an increasing interest in semantic processing which might require a shift of paradigm, possibly reviving the interest in a more theoretical study of language.

The common performance challenges attributed to parsing with hand-crafted grammars concern achieving a balance across efficiency (computational complexity and processing speed), robustness (ability of the parser to produce analysis for any given input) and accuracy (adequacy and correctness of analyses assigned by a parser). The issue of efficiency is related to building a complete parse forest for a sentence during parsing and the implementation of such an ambitious process would be computationally demanding and slow with any type of parsing. However thanks to numerous developments of the last decades parsing with broad-coverage hand-crafted grammars is currently operational and competitive. The issues with coverage are related to difficulties of parsing ungrammatical inputs due to strict restrictions in a hand-crafted lexicon and system of rules, and research of effective methods to reduce and close this gap has produced many fruitful results. In terms of accuracy, grammar-based parsing is capable of producing linguistically precise and sound analyses. A systematic comparison of the performance of state-of-the-art grammar-based and data-driven parsers is complicated by theoretical and structural differences in corresponding analyses, however is necessary for a better understanding of how these frameworks currently relate to each other.

This thesis investigates various aspects of grammar-based and data-driven parsing paradigms using bilexical dependencies as the basis for comparison. The notion of bilexical dependency is central in dependency-based theories of syntax and has lately been widely adopted as a parsing representation in the field. The choice of bilexical dependencies as a basis for comparison in the current work is motivated by their expressivity to directly represent predicate-argument relations (such as subject and object), the existence of transformational rules from phrase structure trees into dependency representations, the central position of the notion of the head in Head-driven Phrase Structure Grammar and practical usage of bilexical dependencies in various applications such as machine translation, sentiment analysis, text summarization, question answering and others. Historically there was no unique standard in dependency representation even for the English language and different research groups manufactured their own formats, such as Prague Dependency (Hajič, 1998) with a long tradition rooted in Functional Generative Description (Sgall et al., 1986) and Stanford Dependencies (de Marneffe et al., 2006) inspired by Lexical-Functional Grammar (Kaplan and Bresnan, 1982). One question that remains open is to what degree the differences in various representations are contentful and significant in parsing, as well as for downstream applications.

Our working environment is the LinGO English Resource Grammar (ERG) (Flickinger, 2000) and the HPSG-based parser PET (Callmeier, 2000) which carries out the construction of a meaning representation in the form of Minimal Recursion Semantics (Copestake et al., 2005) in parallel with the derivation of a syntactic analysis according to the grammar. Experiments are carried out on resources annotated with the English Resource Grammar taken from a variety of different domains and genres, including the newspaper text of the Wall Street Journal (Flickinger et al., 2012), Wikipedia fragments (Ytrestøl et al., 2009), transcripts of spontaneous

speech (Oepen et al., 2004) and others. All these open source tools and resources are part of the international Deep Linguistic Processing with HPSG Initiative (DELPH-IN). Within the DELPH-IN partnership, the modeling of natural language in its complexity is grammar-centric, highly-lexicalized and aiming at high precision and broad coverage.

In the present work we seek to compare the performance of the grammar-based approach to purely statistical state-of-the-art frameworks and furthermore to improve the efficiency of the grammar-based parser by combining it with syntactic dependency parsers.

1.1 Research questions

In the present section we introduce several research questions addressed in this thesis.

- *Which abstract commonalities and differences can be identified among various dependency representations?*

Bilexical dependency representations have developed into a prevailing annotation format alongside the phrase structure representations that are more traditional in the study of English syntax. There are a number of studies aiming to contrast phrase structure and dependency representations (Rambow and Joshi, 1994; Xia and Palmer, 2001; Klein and Manning, 2004; Gerdes and Kahane, 2011), however, to the best of our knowledge, there has been little work on differences and commonalities of various dependency formats at the time when this PhD project was launched. We are therefore interested in examining a variety of dependency formats and determining which differences in dependency representations are linguistically contentful or rather of a more technical or superficial nature.

- *How and to what degree can the syntactic and semantic layers of HPSG be expressed in the form of bilexical dependencies?*

The motivation for expressing HPSG analyses in dependency-style representations is to increase accessibility of the DELPH-IN resources and facilitate cross-framework parse evaluation. Labeled bilexical dependencies are useful for applications such as information extraction, machine translation, sentiment analysis and others. In addition, bilexical dependencies offer ease of readability for consumers of the representation which is an important property of user-centered design. The distinction between syntax and semantics is fundamental to the study of language, and in ERG syntactic information is encoded in a form of a derivation tree and semantics is expressed in the Minimal Recursion Semantics formalism (Copestake et al., 2005). Immediate applications of dependencies representing the syntactic and semantic levels of the ERG analysis are statistical syntactic dependency parsing and statistical semantic dependency parsing, correspondingly.

- *How does the choice of syntactic dependency annotation format affect the performance of dependency parsers and downstream applications?*

The choice of dependency representation touches upon issues that are relevant for dependency parsing and downstream applications. Intrinsic evaluation characterizes parsing performance

mainly with respect to a pre-defined gold standard while extrinsic evaluation assesses the contribution of a syntactic analysis to a certain application. It can be helpful to understand how easy it is to automatically produce different dependency representations, which linguistic phenomena are particularly hard for automatic processing, whether differences in syntactic dependency representations have significant effects on parsing performance and practical tasks, and whether the results of intrinsic and extrinsic evaluation of dependency formats correlate.

- *How does the performance of the HPSG parser relate to data-driven syntactic analyzers?*

With a goal to contribute to our understanding of different approaches to parsing, we are interested in comparing representatives of three different parsing frameworks: phrase structure, dependency and HPSG parsers. The three considered approaches to parsing make different assumptions and produce different output structures which complicates the comparison between them.

- *How can the performance of the HPSG parser be improved by reducing its search space with a native dependency parser?*

The main challenge of parsing with broad-coverage precision grammars is achieving a good balance between efficiency, coverage and accuracy. When optimizing a grammar-based parser for one of these conflicting criteria it is important to account for negative effects in terms of the other two criteria, and for this reason such a study should convey a three-way trade-off evaluation. To address this research question one has to explore a large space of parser integration setups and tackle the problem of choosing high-quality dependencies, and there is little pre-existing knowledge on relevant trade-offs for this particular parser combination problem.

1.2 Contributions

This project can be seen as an attempt to identify and combine the strengths of two paradigms of automatic syntactic analysis: parsing with a hand-crafted grammar grounded in linguistic theory developed over decades by professional linguists, and data-driven statistical parsing implementing modern machine learning algorithms for probabilistic modeling of language from annotated resources. To achieve this goal we extract a dependency backbone from HPSG structures, train statistical dependency parsers on the resultant structures and restrict the search space of the grammar-based parser with the output of the dependency parsers. There are several independent steps involved in this process and below we highlight the main contributions of the present work.

Dependency representations from the ERG analysis layers We develop a fully automated deterministic reduction procedure for transforming internal ERG structures to bilexical dependencies, introducing two novel DELPH-IN dependency schemes. Syntactic dependencies, which we abbreviate DT, are derived directly from the ERG derivation tree, while for semantic dependencies, which we abbreviate DM, a two-step process is implemented for transformation from the native Minimal Recursion Semantics structures into bilexical dependency graphs.

We chose to avoid conversion to existing dependency formats as that involves heuristics and/or rules that modify native properties of the grammar. The use of these conversion heuristics would have introduced confusion in our comparison of the PET parser to data-driven analyzers by making it unclear whether the results of evaluation describe properties of the parser or the converter, and in this light the DT format can be seen as a tool for cross-framework comparison. Another reason why we restrained from conversion to an existing format is that at the start of this PhD project there was no consensus in the field about a syntactic dependency representation standard, something which later motivated the emergence of Universal Stanford Dependencies (de Marneffe et al., 2014) and Universal Dependencies (Nivre, 2015), an attempt to generalize the Stanford dependency format across languages and stimulate the establishment of a unified, cross-linguistic de facto standard.

With an opportunity to derive dependencies from the rich structures of the English Resource Grammar, semi-automatically annotated resources, such as Redwoods (Oepen et al., 2004) and WeScience (Ytrestøl et al., 2009), created within the DELPH-IN partnership become available for direct usage by a broader community in the field. In fact, our semantic dependency format DM has already found an application beyond the present project and was exploited in the SemEval shared tasks of 2014 and 2015 on broad-coverage semantic dependency parsing (Oepen et al., 2014, 2015).

Analysis of dependency representations We performed an in-depth analysis of structural differences and commonalities of a collection of different syntactic and semantic dependency representations, including our two novel DELPH-IN schemes deduced from the ERG. In a qualitative analysis we examine the variations in choices of head and dependents, graph structure and connectivity and discover surprising disagreement among formats for some of the basic syntactic constructions. Quantitative analysis, on the other hand, is interesting for exploring overlap between formats by measuring similarity scores between various formats which provides information about the possibility of interconversion. The syntactic dependency formats are compared on the tasks of dependency parsing and negation resolution. Important findings are that the choice between Stanford Basic (de Marneffe et al., 2006), CoNLL (Johansson and Nugues, 2007) and DT dependencies does not significantly influence parsing performance for state-of-the-art data-driven systems, and that the three schemes are comparable in terms of their utility to the downstream application of negation resolution.

Cross-framework parser comparison Since the structural differences between the three aforementioned representations do not strongly affect syntactic parsing, DT offers a legitimate candidate format for cross-framework parser comparison. In a set of experiments, we find that the grammar-based parser PET shows higher accuracy and better cross-domain resilience than the data-driven direct dependency B&N parser (Bohnet and Nivre, 2012) and PCFG Berkeley parser (Petrov et al., 2006). We carry out several control experiments to ensure that this result is not due to the choice of specific tokenization or part-of-speech tagging convention. The efficiency of PET is similar to the efficiency of B&N which has state-of-the-art accuracy but is slower than most of the other dependency parsers. Unlike dependency parsers, both PET and Berkeley have incomplete coverage although PET to a larger degree. The HPSG and dependency parsers are extrinsically compared on two tasks that use syntactic dependency features:

negation resolution and semantic dependency parsing. We find that negation resolution is not sensitive to the differences in parsing results, while the results of the second task are more accurate when employing the grammar-based parser.

Parser combination We investigate a range of parser combination setups for improved efficiency of HPSG-based parsing via a new interface incorporated in the English Resource Grammar version 1212. We explore several methods for selecting high-quality bilexical dependencies from the output of statistical dependency parsers: a) filtering by part-of-speech of dependent, dependency type and length of dependency span; b) thresholding per-dependency confidence values; c) training an ensemble of parsers with a voting approach. We propose “static” analysis as a method to evaluate the quality of selected dependencies before actual integration with the PET parser relying on precision and annotation rate metrics. On the test sets we discover that a domain-resilient balance across evaluation metrics is achieved when the PET parser is restricted with dependencies selected from the filtered output of parser ensembles with strict voting.

1.3 Thesis outline

This section provides a chapter-by-chapter guide of the dissertation. The thesis structure follows a coherent line of experimental research of the present PhD project.

Chapter 2: Background In this chapter we discuss three syntactic theories that play an important role in contemporary parsing technology. Then we turn to specific parsing algorithms and software examples of constituency, dependency (transition-based and graph-based) and deep grammar parsers. Further we introduce some components of the DELPH-IN computational framework such as the English Resource Grammar (Flickinger, 2000), Minimal Recursion Semantics (Copestake et al., 2005) and Elementary Dependency Structures (Oepen and Lønning, 2006). We review a number of approaches for parser combination and parser evaluation proposed in previous work and describe several linguistic resources that are used in the present work.

Chapter 3: Syntactico-semantic dependencies In this chapter we briefly discuss a usage of bilexical dependencies in such applications as machine translation, semantic search and ontology learning and then motivate the decision to extract dependency representations from the English Resource Grammar. To begin with, we describe our own procedure of reducing the rich HPSG structures to syntactic and semantic dependency graphs, dubbed DT and DM correspondingly. We then carry out a contrasting analysis of nine syntactico-semantic dependency formats examining common linguistic phenomena and measuring the relatedness of the formats using Jaccard similarity and unlabeled F1 score.

Chapter 4: Contrasting parsing experiments Following the structural analysis of dependency representations, we investigate how the properties of syntactic dependencies influence parser results as previous work shows that dependency annotation format may have a significant impact on the parser accuracy. We are contrasting parsing Stanford Basic (de Marneffe

et al., 2006), CoNLL (Johansson and Nugues, 2007), DT and Prague dependencies from the analytical layer of representation (Hajič, 1998). The results are broken down by parser, dependency format, tag set, and in addition we evaluate the effects of including punctuation into scoring and the availability of more training data. In an error analysis we focus on accuracy of the most frequently occurring part-of-speech tags and we exemplify common problems in parsing coordination conjunctions and verbs. In an extrinsic evaluation of the dependency formats on a negation resolution task we get a range of performance differences across different evaluation metrics. An important conclusion of the chapter for the current project is that DT is a good candidate representation for parsing and applications relying on syntactic dependency parsing and that DT compares favorably to the Stanford Basic and CoNLL formats.

Chapter 5: Cross-framework parser evaluation Using DT as a representation for cross-framework parser comparison, we contrast data-driven dependency, statistical phrase structure and HPSG-based parsing architectures—B&N (Bohnet and Nivre, 2012), Berkeley (Petrov et al., 2006) and PET (Callmeier, 2000)—evaluating trade-offs in terms of accuracy, coverage and efficiency on in- and out-of-domain data. The grammar-based parser is available in two configurations: a standard setup optimized for accuracy and a setup optimized for efficiency from Dridan (2013a), with the latter one giving a speed-up of almost a factor of six and achieving parsing speed similar to the dependency parser of Bohnet and Nivre (2012). In our experiments, the grammar-based parser shows significantly higher accuracy and better resilience to domain variation than the two purely statistical systems, but it fails to achieve complete coverage. The error analysis investigates the accuracy of individual dependency types, dependency accuracy relative to dependency length, and distribution of labeled accuracy scores over different lexical categories.

The dependency and HPSG parsers are further compared in two applications that exploit rich syntactic features: negation resolution and semantic dependency parsing. According to the results of our experiments, the first task appears to be insensitive to the choice between HPSG-based and dependency parsers, at least in our setup. The results of the second task indicate that (a) the correlation of syntactic and semantic dependency schemes are important in semantic parsing (e.g. the results are best if syntactic and semantic annotation formats are derived from the same treebank) and (b) grammar-based parsers are very competitive with the purely statistical systems on the semantic annotation formats DM and Enju predicate-argument structures (Miyao and Tsujii, 2005).

Chapter 6: Parser combination The objective of this chapter is to investigate whether the efficiency of the grammar-based parser can be improved without significant loss of accuracy and coverage by its combination with dependency parsers. In the beginning of the chapter we review methods for improving efficiency, coverage and accuracy of HPSG parsing proposed in literature. Subsequently, we move on to discuss experiments on HPSG parsing with dependency constraints. In general, syntactic rules incompatible with enforced dependency relations fail in unification which leads to the reduced search space of the HPSG parser.

One of the main challenges is to select only high-quality dependency relations for integration, and we use several filtering techniques, probability scores of individual parsers and voting in parser ensembles to tackle this problem. We present a series of experiments on the develop-

ment set aiming to choose the best potential configurations and further test the chosen combined setups on in- and out-of-domain sets. Combining parsing ensembles with an HPSG parser we obtain improved efficiency and good coverage without loss of accuracy.

Chapter 7: Concluding remarks The final chapter summarizes the main results of the project, describes some possible refinements and extensions of the presented approaches and outlines directions for the future work experimentation.

1.4 Publications

Parts of this dissertation are based on the following publications. Footnotes at the beginning of the chapters point out which publications are relevant for the respective chapter.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. Who Did What to Whom? A Contrastive Study of Syntacto-Semantic Dependencies. In *Proceedings of the ACL 2012 Sixth Linguistic Annotation Workshop*. Jeju, South Korea, 2012.

Angelina Ivanova, Stephan Oepen and Lilja Øvrelid. Survey on parsing three dependency representations for English. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*. Sofia, Bulgaria, 2013.

Angelina Ivanova, Stephan Oepen, Rebecca Dridan, Dan Flickinger and Lilja Øvrelid. On Different Approaches to Syntactic Analysis Into Bi-Lexical Dependencies An Empirical Comparison of Direct, PCFG-Based, and HPSG-Based Parsers. In *Proceedings of the 13th International Conference on Parsing Technologies (IWPT)*. Nara, Japan, 2013.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova and Yi Zhang. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland, 2014.

Angelina Ivanova, Stephan Oepen, Rebecca Dridan, Dan Flickinger, Lilja Øvrelid and Emanuele Lapponi. On Different Approaches to Syntactic Analysis Into Bi-Lexical Dependencies An Empirical Comparison of Direct, PCFG-Based, and HPSG-Based Parsers. *Journal of Language Modelling*. In press.

Chapter 2

Background

The goal of this chapter is to introduce the relevant theoretical background, to equip the reader with the necessary terminology and give an overview of the origins of the challenges that are addressed in the present work.

First we will briefly review three linguistic theories that constitute a foundation for the data representations and software tools exploited in our experiments, as well as explain important linguistic notions which we will use in subsequent chapters. Thereupon we will launch the discussion of the core topic of this thesis—automatic syntactic analysis, e.g. parsing. We provide a high-level description of both data-driven and grammar-based approaches to parsing and give examples of parsers representing the different frameworks. Further on we will describe various approaches to parser combination, summarize a range of evaluation metrics used in parser comparisons and introduce several important linguistic resources that will be central in the remaining chapters of the thesis.

2.1 Syntactic theories

The word *syntax* derives from ancient Greek *συνταξις* (“arrangement”) and refers to the arrangement of words in sentences, e.g. the structure and formation of sentences and the relationship of their component parts. The first grammar developments originated in ancient India and ancient Greece. One of the oldest prominent grammar studies is Pāṇini’s theoretical analysis of Sanskrit known as *Aṣṭādhyāyī* (ca. 350 BC). This grammar formulates morphological, syntactic and phonological rules though this subcategorisation is not clear-cut in the original work. Since then, a range of different theories have been proposed to account for the syntactic properties of natural languages.

This section addresses three influential modern theories of syntax: Phrase-Structure Grammar, Dependency Grammar and Head-Driven Phrase Structure Grammar.

2.1.1 Phrase-structure grammar

The concept of *phrase structure (constituency)* suggests that groups of words may behave as a single unit or constituent, e.g. “a star” is a noun phrase that acts as one entity in an English sentence. Constituency tests, such as topicalization and coordination, allow us to determine whether a sequence of words forms a constituent. Topicalization is a syntactic movement that

places the emphasis on the topic by positioning it in the beginning of the sentence (compare “He saw *a star* with a telescope” and “*A star* he saw with a telescope”). A string of words can be topicalized in a sentence if, and only if, it is a constituent. As a general rule, only constituents (of the same type) can be coordinated in a sentence using coordinating conjunctions (e.g. “He saw *a star* and *the moon* with a telescope”), with an exception of so-called coordination of unlikes (e.g. “He is wealthy and a Republican”). Constituent categories are also motivated by distributional equivalence, e.g. constituents can be identified with substitution tests: if a group of words can be substituted with a pro-form, it is a constituent (e.g. since the substitution “He saw *a star* with a telescope”—“He saw *it* with a telescope” is valid, “a star” is a constituent).

Phrase-structure rules formalize generalizations about the phrase and sentence structure. Some concepts of phrase structure grammar can be traced back to the analysis of logical propositions by ancient Stoics and centuries later passed through formal logic to linguists such as Leonard Bloomfield, Rulon Wells, Zellig Harris, and Noam Chomsky (Covington, 2001).

In 1957, Noam Chomsky published his book on Syntactic Structures (Chomsky, 1957), where he formalized the idea of basing a grammar on constituent structure using *context-free grammar* (CFG) (see below). Since then Chomskyan theories have become a very influential school of thought in linguistics. The theories were developed in the framework of *generative grammar* based on the view that humans have an innate capacity for language and attempting to describe principles and rules sufficient to generate all and only the well-formed sentences of a language.

The common ancestor for phrase structure theories of syntax is Transformational Grammar (Chomsky, 1957, 1965) which augments a lexicon and phrase structure rules expressing “deep structure” of the sentence with transformations that map sentence configurations to the “surface structure” of the sentence. For example, passive voice is a surface structure that can be recovered via a series of transformations of the active form of the sentence representing its deep structure.

Natural language processing implementations of phrase structure grammars departed from direct realizations of later linguistic theories in the Chomskyan tradition, such as Government and Binding (Chomsky, 1981) and the Minimalist Program (Chomsky, 1995), comprising only some aspects of these theoretical views like, for example, the formal model of CFG (Chomsky, 1957) and some of the notions of X’ theory (Chomsky, 1970). Below we will introduce the notions from these theories that have had an important influence in computational linguistics.

Context-Free Grammar (CFG)

Context-Free Grammar (CFG) (Chomsky, 1957) is a formalism for modeling constituent structure in natural languages which is defined as a quadruple $G = (N, \Sigma, R, S)$ (Jurafsky and Martin, 2009, p.425):

- C is a set of *non-terminal symbols* (syntactic categories);
- Σ is a set of *terminal symbols* (lexical categories);
- R is a set of *productions*, or *rules*;
- $S \in C$ is a designated symbol called the *initial symbol*.

The productions formally explain generalizations about the phrase and sentence structure and are denoted as $A \rightarrow \beta$ where A is a non-terminal and β is a string of non-terminals and/or terminals. A production of the form $A \rightarrow \varepsilon$ is called an *epsilon rule*, or *null rule*.

A string A *derives* a string β if A can be rewritten as β by a sequence of rule applications (Hopcroft and Ullman, 1979):

- if $A \rightarrow \beta \in R$ and $\alpha, \gamma \in (\Sigma \cup C)^*$ then $\alpha A \gamma \Rightarrow \alpha \beta \gamma$ ($\alpha A \gamma$ derives $\alpha \beta \gamma$);
- if $\alpha_1, \alpha_2, \dots, \alpha_n \in (\Sigma \cup C)^*$ and $\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \dots, \alpha_{n-1} \Rightarrow \alpha_n$, then $\alpha_1 \xRightarrow{*} \alpha_n$ (α_1 derives α_n).

A context-free grammar G defines a formal language $\mathcal{L}(G) = \{w \in \Sigma^* | S \xRightarrow{*} w\}$, i.e. if a string of words can be derived from the initial symbol by sequential applications of the grammar rules, it is licensed by the grammar as a valid sentence of the language $\mathcal{L}(G)$. Strings that cannot be derived are considered ungrammatical. Grammars that define a language by the set of possible sentences “generated” by the grammar are called generative.

A toy example of a CFG is shown in Example (1); productions on the right are called *lexical rules* as they do not contain non-terminal symbols on the right-hand side.

(1)	$S \rightarrow NP VP$ $NP \rightarrow PRP$ $NP \rightarrow DT N$ $NP \rightarrow DT N PP$ $VP \rightarrow V NP$ $VP \rightarrow V NP PP$ $PP \rightarrow P NP$	$PRP \rightarrow He$ $V \rightarrow saw$ $DT \rightarrow a$ $N \rightarrow star$ $N \rightarrow telescope$ $P \rightarrow with$
-----	--	--

Our grammar may generate multiple analyses for a given sentence due to the lexical or structural ambiguity of a sentence. Typically, syntactic ambiguity, e.g. the availability of multiple distinct derivations, is considered an indicator of semantic ambiguity, i.e. different possible interpretations, as demonstrated in Figure 2.1 with an example of two context-free grammar analyses for the ambiguous sentence “He saw a star with a telescope”. The first syntactic structure suggests that the (movie?) star was holding the telescope, while the second interpretation indicates that the telescope was used to see the star.

Probabilistic Context-Free Grammar (PCFG)

Probabilistic Context-Free Grammar (PCFG) first proposed by Booth (1969), is a probabilistic version of a CFG. The formal definition of a PCFG differs from the definition of a CFG presented above only by the fact that all productions are assigned some probability p :

$$A \rightarrow \alpha [p]$$

p is the conditional probability of the right-hand-side string α given the left-hand-side non-terminal A . Probabilities of all productions of a non-terminal must sum up to one:

$$\sum_{\alpha} p(A \rightarrow \alpha) = 1$$

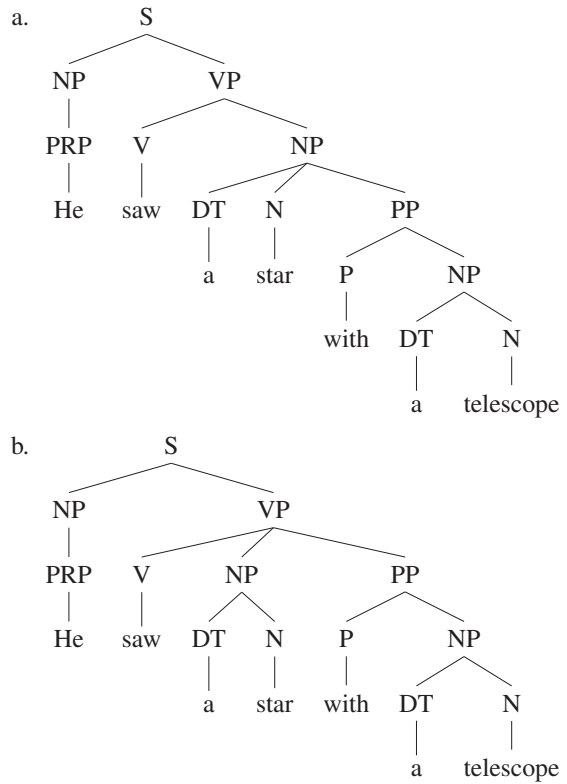


Figure 2.1: Two syntactic analyses for the sentence “He saw a star with a telescope”

The assumption of the grammar is that the rules are independent of the context, therefore the joint probability of a tree T and a sentence S is the product of the probabilities of the rules used to build the tree:

$$p(T, S) = \prod_{r \in T} p(r)$$

The probability of the sentence w equals to the sum of probabilities of all possible parse trees that could generate this sentence:

$$p(w) = \sum_{T: s(T)=w} p(T, S), \text{ where } s(T) \text{ is the terminal yield of } T$$

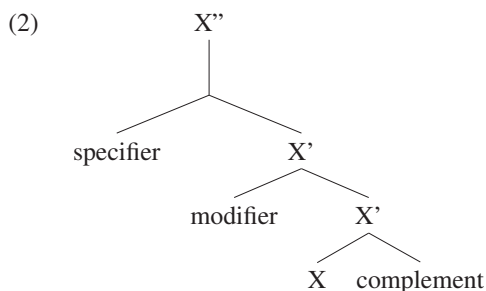
A PCFG is consistent if and only if the sum of the probabilities of all sentences in the language is one (Wetherell, 1980):

$$\sum_{w \in L(G)} p(w) = 1$$

In practice the probabilities of productions are empirically approximated from a corpus.

X' theory

X' theory (Chomsky, 1970) introduces abstract patterns characterizing internal phrasal and sentential structures in a language. The main syntactic properties of the phrase are centered in the head and the phrase is said to be a projection of the head. Most commonly two levels of projection are employed. One of the versions of the X'-scheme for English is shown in Example (2):



where X is the head, X' is a semi-phrasal intermediate projection of X and X'' is the maximal projection of the head. V-VP-S and N-N'-NP are common instances of this schematic version.

The *modifier* is an optional word or phrase that limits or changes the sense of its head (another word or phrase). A modifier appearing before its head, e.g. “What a *beautiful* day!”, is called a premodifier, and a modifier in position after the head, e.g. “a girl *in the blue dress*”, is called a postmodifier. Modifiers cause recursive X' nodes to appear.

The *complement* is a word or phrase that completes the sense of its predicate, e.g. “He has accepted *my apology*”.

The *specifier* is defined as a non-recursive left daughter of the phrasal node. Determiners are NP specifiers, e.g. “*the* boy”. A specifier at a sentence level is subject. A head can have several modifiers and complements, but only one specifier.

The theory abstracts from the linear order of the constituents at each level and constrains only the hierarchical structure. X' theory was integrated in Government and Binding system of theories and concerned with the representation of sentences as hierarchical structures in the form of tree diagrams.

Generalized Phrase-Structure Grammar (GPSG)

Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985) is a theory that describes the syntax of natural languages in the framework of context-free grammars. The mathematical properties of this theory were the focus of the research and the grammar was thoroughly formalized whereby there emerged parsing systems built on this formalism (Ramsay, 1985; Devos and Gilloux, 1990). GPSG had a direct influence on the development of the HPSG theory (Pollard and Sag, 1994) that we are going to introduce in this chapter and that will be central in this thesis.

Whereas Transformational Grammar and Government and Binding assume that sentences have two levels of syntactic representation, deep and surface structures, GPSG posits only one level which is closer to surface structure, therefore GPSG, and for the same reason HPSG, are called “monostratal” theories. The effects of transformations are achieved by capturing long-distance dependencies in complex features and using metarules for generalization instead of using the movement rules (Horáček et al., 2011a). The feature-based analysis of filler-gap constructions (such as topicalization, *wh*-questions, and relative clauses) is retained in the HPSG approach. Context-free grammar rules are divided into rules of immediate dominance (“ID rules”) and rules of linear precedence (“LP rules”) which are also employed in HPSG (Sag et al., 2003)[p. 536].

The theory introduces the operation of feature unification, which is similar to the set union operation but undefined for the case when features contradict each other (Horáček et al., 2011a). Phrases are considered to have a head, which is a category-defining element, and a foot, which is a complement, and the grammar imposes the head and foot principles. The head principle declares that the head features of a child node in a tree must be identical to the head features of the parent node (Horáček et al., 2011a). The foot principle states that the foot features instantiated on a parent category in a tree must be identical to the unification of the foot features in all its child nodes (Horáček et al., 2011a).

2.1.2 Dependency grammar

In contrast to a phrase structure grammar that employs constituents as its main building blocks, a dependency grammar highlights lexical items. While in a phrase structure grammar sentences are built up from constituents, in a dependency grammar sentence structure is based on dependency relations between lexical items.

A *dependency relation* holds between a *head (governor)* and its *dependent(s) (modifier(s))*. The intuition that the notion “head” conveys is that one element characterizes or dominates the whole unit in a syntactic structure (Zwicky, 1985). Some common criteria for identifying heads and dependents are the following (Zwicky, 1985; Hudson, 1990; Nivre, 2005):

- the head can replace the syntactic construction that it governs;



Figure 2.2: Constituency and dependency analysis of the sentence “Mike plays football”

- the head bears the semantic category and is specified by the dependents;
- the head is mandatory and it determines whether dependents are obligatory or optional;
- grammatical categories of dependents are defined by the head via agreement or government;
- the head specifies the linear order of its dependents.

Dependency relations are binary asymmetric relations between lexical items (Kübler et al., 2009)[p. 2]. Every dependency relation bears a syntactic function which can be explicitly expressed via a dependency label. Dependency relations often express grammatical relations that refer to syntactic functions of the constituents in a clause, e.g. subject, direct object, indirect object, attribute, complement, specifier, determiner, modifier and others.

In 1959, the book “Elements of Structural Syntax” of the French linguist Lucien Tesnière was published posthumously introducing his theory of syntax that later became known as *dependency grammar*. Tesnière (1959) emphasizes that components of the sentence are words that are connected by syntactic relations. For example, the phrase “John slept” consists of the words “John”, “slept” and the syntactic relation that joins the two elements and expresses the fact that it was John who slept.

Tesnière argued for the autonomy of syntax from morphology and semantics, based in part on the observations that:

- syntax investigates the inner form of the sentence, conceptually different from the outer form of the sentence studied by morphology and both obey their own rules;
- there are semantically absurd sentences that are structurally correct.

Tesnière proposed that dependency grammars are verb-centered (e.g. a finite verb often functions as head in binary relations and often appears at the root of a clause) unlike constituency grammars that suggest a subject-predicate division of the clause $S \rightarrow NP VP$ (see Figure 2.2).

In the mid-1960s Mel’čuk and Žolkowskij developed the *Meaning-Text Theory (MTT)* (Žolkowskij and Mel’čuk, 1965, 1967). The authors postulate dependencies to account for

the syntactic analysis within their framework. MTT is a multi-stratal linguistic model that establishes correspondence between meanings and sounds. In MTT, the meaning is “the only invariant property of all possible paraphrases of an utterance” (Mel’čuk, 2012)[p.22], the text is “any linguistically valid segment of speech” (Mel’čuk, 2012)[p.91], the surface form of meaning, and the sentence is “a maximal utterance that consists of clauses and is a complete unit of communication” (Mel’čuk, 2012)[p.31]. The model consists of seven levels of sentence representation with six modules that establish correspondence between the adjacent levels (see Figure 2.3 from Kahane (2003)).

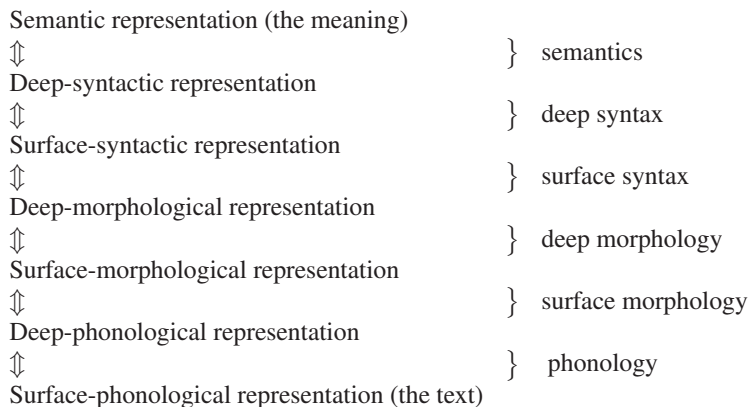


Figure 2.3: Levels of sentence representation and modules of the MTT (Kahane, 2003)

Surface phonology, or phonetics, which establishes the correspondence between the surface-phonological representation (phonetics) and actual sounds is outside of the MTT scope. MTT postulates that a natural language is a logical device that provides many-to-many correspondences between meanings and texts. Synonymy is the phenomenon of one meaning corresponding to many texts and homonymy/polysemy is the phenomenon of one text corresponding to many meanings. The model describes how to synthesise a sentence from a semantic representation and how to do the inverse (analysis), with more emphasis on the synthesis direction under the assumption that the real language knowledge requires the ability of speaking it.

The surface-syntactic structure of a sentence is a tree with all lexemes of the sentence (including all auxiliary words) as nodes and language-specific surface-syntactic relations as arc labels (*syntactic dependencies*).

The deep syntactic structure of a sentence is a tree with the lexemes of the sentences as nodes and universal deep-syntactic relations as arc labels (*syntactic dependencies*). Some lexical units of the sentence (auxiliaries, substitute pronouns, governed prepositions and conjunctions) are not represented in this tree while there are additional empty (*fictitious*) lexemes exemplifying syntactic constructions that are not realized in the surface form of the original sentence. An example of such fictitious lexeme is “approximately” in noun+numeral constructions in the Russian language, cf. “5 dneĵ” (“5 days”) and “dneĵ 5” (“approximately 5 days”).

The semantic structure of a sentence is a network with nodes representing meanings and arcs expressing predicate–argument relations (*semantic dependencies*).

The MTT theory was successfully implemented in a number of engineering systems, including French-to-Russian and English-to-Russian machine translation systems (Apresjan et al., 1984-85, 1989), and in text generation systems to produce multilingual weather forecasts (Kittredge and Polguère, 1991; Coch, 1998).

Dependency-based theories of syntax did not receive as much attention as phrase structure theories until recently in part due to the publication of Gaifman (1965) where it was shown that i) dependency and context-free grammars describe the same class of languages – context-free languages; ii) for every dependency grammar there exists a corresponding CFG but not vice versa (Nivre, 2005). The conversion from dependency to constituency grammar is thus one-way, as the inverse operation is possible only for a subclass of CFG. However these claims hold only for the restricted variant of dependency grammars formally described in Hays (1964) and Gaifman (1965), whereas in general, dependency grammar is not a subclass of CFG (Neuhaus and Bröker, 1997). One of the restrictions that the Hays and Gaifman grammar had was a *projectivity* property that does not allow dependency arcs to cross (compare Figures 2.4 and 2.5 from Nivre (2008)).

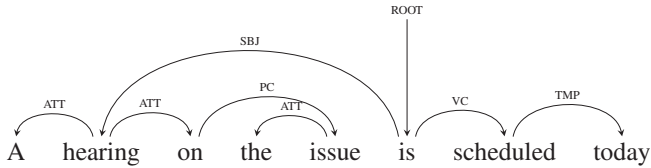


Figure 2.4: Projective dependency tree example from Nivre (2008)

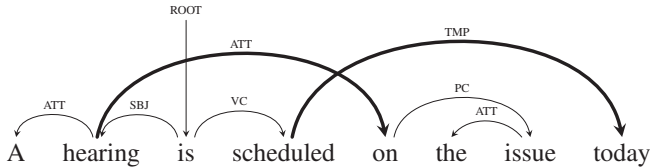


Figure 2.5: Non-projective dependency tree example from Nivre (2008). Dependencies in bold are non-projective.

Dependency grammar is more expressive than CFG because non-projective dependencies cannot be directly conveyed in CFG. Non-projectivity is more common in languages with a more flexible word order than English, such as Russian and German.

For practical purposes dependency representations may be formalized as dependency graphs (Nivre et al., 2007b):

A *dependency graph for a sentence* is a labeled directed graph $G = (V, E, L)$ with the indexed nodes corresponding to the tokens of the sentence $x = (w_1, \dots, w_n)$ where

- $V = (v_0, v_1, \dots, v_n)$ is the set of indexed nodes with the indexes from 0 to n ;
- $E \subseteq V \times V$ is the set of arcs connecting pairs of nodes (v_i, v_j) with the i th node being the head and the j th node its dependent;

- $L : E \rightarrow R$ is a labeling function that assigns types (arc labels) from the set R to the arcs from the set E .

For example, the indexed nodes of the graph in Figure 2.4 correspond to the ordered set of indexed sentence tokens $V = \{\text{ROOT}_0, A_1, \text{hearing}_2, \text{on}_3, \text{the}_4, \text{issue}_5, \text{is}_6, \text{scheduled}_7, \text{today}_8\}$; the set of arcs is $E = \{(\text{ROOT}_0, \text{is}_6, \text{ }), (\text{hearing}_2, A_1), (\text{hearing}_2, \text{on}_3), (\text{on}_3, \text{issue}_5), (\text{issue}_5, \text{the}_4), (\text{is}_6, \text{hearing}_2), (\text{is}_6, \text{scheduled}_7), (\text{scheduled}_7, \text{today}_8)\}$ and the set of arc labels is $R = \{\text{ATT, SBJ, PC, VC, TMP, ROOT}\}$.

Dependency graphs can be ordered or unordered. For example, the Prague Theory (Sgall et al., 1986) uses unordered dependency trees at the deeper level of representation and ordered dependency trees at the surface level. However, most formalizations only support ordered tree representations.

The dependency graph G is well-formed if the following conditions are satisfied (Nivre et al., 2007b):

- the node with the index 0 is a root;
- G is weakly connected;
- every node in G can have at most one head, e.g. if $v_i \rightarrow v_j$ then $\nexists\{k | k \neq i \text{ and } v_k \rightarrow v_j\}$;
- G is acyclic, i.e. if $v_i \rightarrow v_j$ then $v_j \not\stackrel{*}{\rightarrow} v_i$

Both dependency trees in Figure 2.4 and Figure 2.5 are well-formed.

As it has been mentioned above, some variants of dependency grammar have a projectivity constraint:

$$\text{if } v_i \rightarrow v_j \text{ then } i \stackrel{*}{\rightarrow} k, \text{ for every node } k \text{ such that } i < k < j \text{ or } j < k < i.$$

The tree in Figure 2.4 satisfies the projectivity constraint, while the tree in Figure 2.5 does not.

2.1.3 Head-Driven Phrase Structure Grammar

Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) is a lexicalized formal theory of language that encodes syntactic and semantic information in feature structures. HPSG was initially developed as an extension of Generalized Phrase Structure Grammar (see Section 2.1.1). The following introduction will be loosely based on a simplified presentation of HPSG by Sag et al. (2003).

HPSG incorporates linguistic information in *typed feature structures* and embodies generalizations over classes of related objects in a *type hierarchy*. The main components of the grammar are the *lexicon*, *principles* and *rules* and the main operation to check and aggregate information is *unification*. HPSG shares the central position of syntactic heads with dependency-based theories of syntax and the use of phrase-structure rules from constituent-based theories.

Feature structures

A *feature structure* (Kay, 1985; Shieber, 1986; Carpenter, 1992) is a set of feature-value pairs representing grammatical information. It is common to depict them in an *attribute-value*

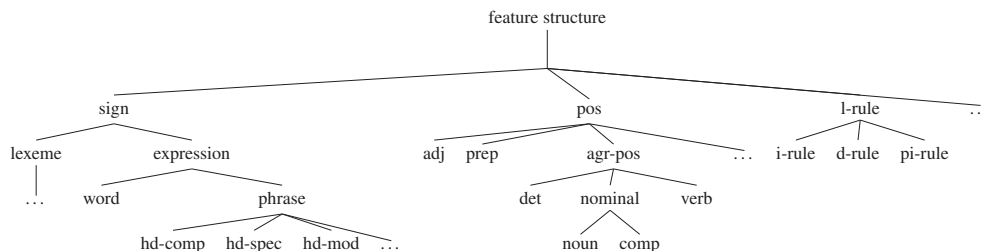


Figure 2.6: The type hierarchy from Sag et al. (2003)

matrix representation as shown in (3), where POS and NUM are features and noun and sg are their corresponding values.

$$(3) \begin{bmatrix} \text{POS noun} \\ \text{NUM sg} \end{bmatrix}$$

Feature structures can be nested because some features take other feature structures as their values. In HPSG, each feature structure is associated with a type that imposes appropriate constraints on the occurrences of features and on the values that they take. More formally, a typed feature structure is a rooted, directed, labeled graph.

Type hierarchy

Linguistic entities are attributed to certain classes, called grammatical types, that are associated with features. The types are organized in a multiple-inheritance hierarchy like the one partially presented in Figure 2.6 (Sag et al., 2003) [p. 492].

HPSG is related to the sign-based conception of Ferdinand de Saussure: words and phrases of the language are represented as *signs*. Saussure suggested that a sign establishes an association of a mental representation of a sound with a mental representation of a meaning. In HPSG a sign is the most general type of feature structure that includes phonological, syntactic and semantic information (Sag et al., 2003) [p. 475]. *Lexeme* is an abstract class of all forms of a word, e.g. {go, goes, going, went, gone} for the lexeme “go”. *Words* and *phrases* are subtypes of the class *expression* which expresses the intuition of their common properties, such as the feature HEAD with part-of-speech value, verb inflection etc. Examples (4) and (5) illustrate a feature structure representation of the lexical entry for a noun and of the category NP.

$$(4) \begin{bmatrix} \text{word} \\ \text{HEAD noun} \end{bmatrix}$$

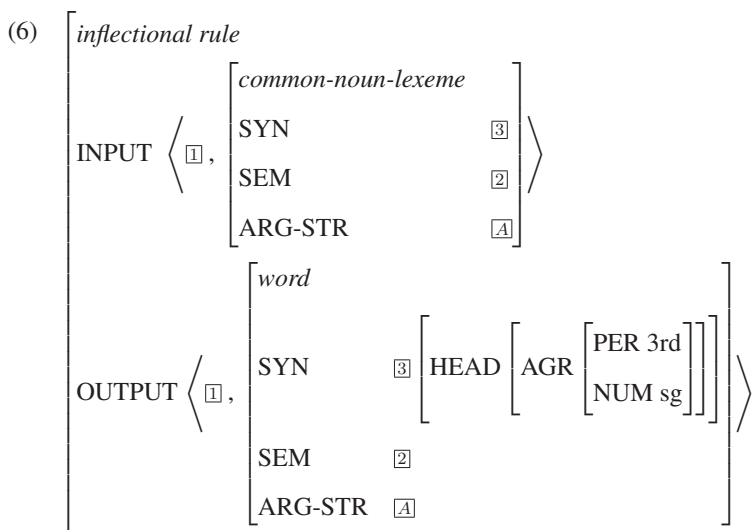
$$(5) \begin{bmatrix} \text{phrase} \\ \text{HEAD noun} \end{bmatrix}$$

Part-of-speech types, or *lexical categories*, form a hierarchy with respect to the features that they share: for example, nouns, verbs and determiners take the feature AGR (“agreement”).

Agreement marks that e.g. subject and verb agree in person and number. Figure 2.6 shows that the feature AGR is introduced by a subtype of part-of-speech (*agr-pos*), which provides the common supertype for the parts of speech determiner (*det*), nominal and verb.

Lexical rules (l-rules) establish relations between input and output values. The lexical rules are expressed in the form of feature structures and constrain their input and output to have the same semantic values. *Inflectional rules* (i-rules) are a special case of lexical rules that establish the conversion of lexemes into words (“realizations” of lexemes), preserving the same semantic (SEM), syntactic (SYN) and argument-structure (ARG-STR) (the subject, the complements etc.) values. *Argument-structure* is an ordered list of arguments required by the sign.

The feature structure in Example (6) from Sag et al. (2003) [p. 253] shows a singular noun inflectional rule that transforms a given lexeme to a singular noun word. Values in boxes are used as variables to indicate identity of values. Inflectional rules can also attach affixes, e.g. nominal plural and verbal past tense. In fact, a singular noun can be viewed as using an empty suffix. In addition, there are derivational rules with empty suffixes such as verbification.



Another type of lexical rules are *derivational rules* that attach prefixes or suffixes to the input lexemes or tackle the problem of valence alternations, such as dative alternation in Example (7) from Levin (1993)[p.46]:

- (7) Bill sold a car to Tom.
 Bill sold Tom a car.

Derivational rules output lexemes that can be further processed by inflectional rules. Example (8) from Sag et al. (2003) [p. 260] illustrates an agent nominalization lexical rule (for transformations like “teach” - “teacher”):

$$(8) \left[\begin{array}{l} \text{derivational rule} \\ \text{INPUT} \left\langle \textcircled{2}, \left[\begin{array}{l} \text{strict-transitive-verb-lexeme} \\ \text{SEM} \quad \left[\text{INDEX } s \right] \\ \text{ARG-STR} \quad \langle X_i, NP_j \rangle \end{array} \right] \right\rangle \\ \text{OUTPUT} \left\langle F_{\text{-er}}(\textcircled{2}), \left[\begin{array}{l} \text{count-noun-lexeme} \\ \text{SEM} \quad \left[\text{INDEX } i \right] \\ \text{ARG-STR} \quad \left\langle Y, \left(\begin{array}{l} PP_j \\ [FORM \text{ of}] \end{array} \right) \right\rangle \end{array} \right] \right\rangle \end{array} \right]$$

The function $F_{\text{-er}}$ adds a suffix “er” to the lexeme of the strictly transitive verbs (ambitransitive verbs such as “leave” are filtered out by this restriction) and outputs the lexeme of the countable noun type. The new lexeme’s index (i) matches the index of the subject of the original lexeme (X_i); the output agent nominal takes a determiner (Y) and a PP complement that matches the object of the input verb (j): “to teach mathematics” - “teacher of mathematics”.

Lexicon

The lexicon plays a central role in HPSG. The grammar is lexicalized meaning that the most linguistic knowledge is encoded in lexicon entries. Without means of generalization there would be a great deal of redundant syntactic and morphological information in individual lexical entries. Mechanisms expressing lexical regularities in HPSG, initially elaborated in Flickinger (1987), are categorization of lexical items in a hierarchy with class inheritance of syntactic properties and introduction of lexical rules for morphological feature manipulations.

HPSG principles

The two core principles of HPSG are the head feature principle and the valence principle.

The Head Feature Principle: the HEAD value of the phrase is equal to the HEAD value of its head daughter.

The Valence Principle: valence requirements of a phrase (specifier and complement constraints) are identified by the head daughter.

The term *valence*, borrowed from chemistry, describes the capacity of a head to take a certain number of arguments of a specific type. These principles also underline the essential role of the syntactic head in HPSG.

HPSG constructions

The main grammar constructions of HPSG are based on three major grammatical functions: modifier, complement and specifier introduced above in the framework of X’ theory. Sag et al. (2003)[p.64] introduce complement and specifier as generalizations of the notions of object and determiner respectively. Among the principal grammar constructions that the lexical items satisfy in valid sentences of a language are the head-modifier, the head-complement and the

head-specifier rules. SPR and COMPS lists of HPSG signs are used to allow a head to require specific types of specifiers and complements respectively. A MOD list is used to enable modifier to select the head it can attach to. These lists are shortened once the appropriate arguments have been found.

The Head-Modifier Construction (see Example (9)): *A phrase can consist of a lexical or phrasal head followed by a compatible modifier phrase* (Sag et al., 2003)[p.502].

$$(9) \text{ [phrase]} \rightarrow \mathbf{H}_{\boxed{1}} \left[\begin{array}{l} \text{COMPS} \quad \langle \rangle \\ \text{MOD} \quad \langle \boxed{1} \rangle \end{array} \right]$$

Examples of head-modifier and modifier-head constructions (the head in bold, the modifier in italics): the **house** *on the hill*, a new **house**.

The Head-Complement Construction (see Example (10)): *A phrase can consist of a lexical head followed by all its complements* (Sag et al., 2003)[p.502].

$$(10) \left[\begin{array}{l} \textit{phrase} \\ \text{VAL} \quad \left[\text{COMPS} \quad \langle \rangle \right] \end{array} \right] \rightarrow \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{VAL} \quad \left[\text{COMPS} \quad \langle \boxed{1}, \dots, \boxed{n} \rangle \right] \end{array} \right] \boxed{1} \dots \boxed{n}$$

Example of the head-complement construction (the head in bold, the complement in italics): Paul **watched** *a movie*, **in** *the hills*.

The Head-Specifier Construction (see Example (11)): *A phrase can consist of a lexical or phrasal head preceded by its specifier* (Sag et al., 2003)[p.501].

$$(11) \left[\begin{array}{l} \textit{phrase} \\ \text{SPR} \quad \langle \rangle \end{array} \right] \rightarrow \boxed{1} \mathbf{H} \left[\begin{array}{l} \text{VAL} \quad \left[\begin{array}{l} \text{SPR} \quad \langle \boxed{1} \rangle \\ \text{COMPS} \quad \langle \rangle \end{array} \right] \end{array} \right]$$

Examples of specifier-head constructions (the head in bold, the specifier in italics): *this* **bag**, *the* **match**, a **boy**.

The Head-Subject Construction (see Example (12)): *combines a head such as a verb phrase with its subject* (Tojo and Saito, 2005).

$$(12) \left[\begin{array}{l} \textit{phrase} \\ \text{SUBJ} \quad \langle \rangle \end{array} \right] \rightarrow \boxed{1} [] \mathbf{H} \left[\begin{array}{l} \textit{word} \\ \text{SUBJ} \quad \langle \boxed{1} \rangle \end{array} \right]$$

Examples of subject-head constructions (the head in bold, the subject in italics): *Jane* **arrived**, *Bob* **sneezed**.

Unification

Valid expressions of the language defined by the *grammar principles* (such as the head feature and valence principles described above) and *grammar constructions* are verified via *unification*. Unification is an operation equivalent to conjunction in logic, which simply combines

constraints of two feature structures. Unification holds only if the information in two signs is consistent, therefore grammaticality of a sentence can be checked via unification of particular signs of a sentence with feature structures of the HPSG constructions. We will depict unification with the symbol \sqcup . Example (13) (Horáček et al., 2011b) shows that the property “singular number” can be unified with the property “3rd person” but not with “plural”.

$$(13) \quad \begin{array}{l} \left[\text{NUM sg} \right] \sqcup \left[\text{PER 3rd} \right] = \begin{bmatrix} \text{NUM sg} \\ \text{PER 3rd} \end{bmatrix} \\ \left[\text{NUM sg} \right] \sqcup \left[\text{NUM pl} \right] = \text{fail} \end{array}$$

Subsumption

A feature structure L subsumes a feature structure M when the latter contains more constraints than the former, e.g. a more abstract feature structure is said to subsume a more specific one (see Example (14) (Wintner, 2005), where subsumption is denoted with a symbol \sqsubseteq). Not all feature structures can be compared under subsumption though (e.g. M and N in Example (14) are incompatible).

$$(14) \quad \begin{array}{l} \text{L: } \left[\text{PERS 3} \right], \quad \text{M: } \begin{bmatrix} \text{PERS 3} \\ \text{NUM sg} \end{bmatrix} \quad \text{N: } \begin{bmatrix} \text{PERS 3} \\ \text{NUM pl} \end{bmatrix} \\ L \sqsubseteq M, L \sqsubseteq N, M \not\sqsubseteq N, N \not\sqsubseteq M \end{array}$$

In conclusion, it is worth mentioning that HPSG is not the only full-fledged linguistically sound grammar formalism to date. Other long-established research lines include such well-defined frameworks as Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) and Combinatory Categorical Grammar (CCG) (Steedman, 2000). These are however not in the focus of the present work.

2.2 Syntactic parsing

Automatic syntactic analysis, or *parsing* of natural language, is the process of automatically producing syntactic structure for an input text. When parsing a sentence, we often seek its underlying logical content, or another formal and abstract representation of sentence meaning, and determining the grammatical structure is a means towards this goal.

In the parsing universe there are two principal approaches that increasingly incline toward each other: data-driven and grammar-based parsing. Data-driven parsing methods rely on machine-learning algorithms in order to learn linguistic information implicitly encoded in texts, that are often annotated with “correct” target grammatical structures. Grammar-based parsing exploits explicit hand-crafted formal grammars. The two approaches are not polarly distinct but rather more and more interrelated. Data-driven methods require treebanks annotated with linguistic information and with sophisticated statistical algorithms they can learn expressive grammars, so-called treebank grammars (Charniak, 1996). As we have discussed, there are also grammar-less data-driven methods (such as the MaltParser (Nivre et al., 2007b)). Some data-driven are built upon linguistically rich features and representations (Klein and Manning, 2003;

Bod, 1998; Øvrelid and Nivre, 2007). In turn, grammar-based systems rely on statistical techniques for disambiguation and often also for increased robustness and the acquisition of lexical resources. In grammar-based parsing the analysis is constrained by the rules and assumptions adopted by the grammar. Data-driven parsing is heavily influenced by the frequencies of linguistic phenomena in texts selected for training.

Bunt et al. (2010) highlight among the main trends related to parsing: 1) a shift of focus from traditional context-free grammars to dependency grammars; 2) developing models with latent variables. The first trend has led to a variety of dependency formalisms, parsers and representations being designed. Examples of the second trend embodiment are the PCFG model with latent annotations of Matsuzaki et al. (2005) and the Berkeley parser of (Petrov et al., 2006) which relies on latent variables in order to derive a grammar from the data by assigning latent variables to each nonterminal node of PCFG and estimating their parameters with the EM algorithm.

2.2.1 Data-driven parsing

Data-driven parsers rely on syntactically annotated corpora, or treebanks, in order to construct tree-structure analyses of texts. A majority of such parsers do not utilize hand-crafted grammars at all but derive linguistic knowledge from the empirical data. In the following sections we will introduce constituent parsing with the Berkeley parser as an illustrative example, transition-based dependency parsing with the Malt parser as an illustration, graph-based dependency parsing with the MST parser as the software that represents the approach and the hybrid system of Bohnet and Nivre (2012) that combines the advantages of transition- and graph-based parsing.

Constituent parsing

Constituent parsing can be described as the search for the right parse tree describing a derivation of the sentence in a (typically context-free) Phrase Structure Grammar among all possible parse trees. A sentence of a language is considered to be recursively composed from smaller segments called constituents or phrases that carry the meaning and the structure on their own. In a bottom-up approach, the parser begins with the input tokens and builds up trees, applying CFG rules until the root node is reached. In a top-down approach a sentence is broken up into constituents (phrases), which are then broken into smaller constituents until the terminal nodes are reached.

Berkeley parser The Berkeley parser (Petrov et al., 2006) learns probabilistic context-free grammars (PCFG) which assign a sequence of words the most likely parse tree. The algorithm starts from a naive PCFG-grammar which consists of empirical rules and probabilities from the treebank and then repeatedly applies a split-and-merge approach for automatic grammar refinement. Splitting assures a fit to the training data, while merging facilitates generalization and controls grammar size. The splits are guided by the EM algorithm which is based on the likelihood of the training trees. Complex and frequent categories such as NP and VP have the

most diverse set of subcategories while rare or simple ones like functional categories generally show fewer splits. Merging is used to prevent oversplitting the grammar.

Transition-based vs. graph-based dependency parsing The two dominating approaches to data-driven dependency parsing are the transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2004) and graph-based (McDonald et al., 2005a) approaches. Transition-based algorithms make parsing decisions by scoring individual transitions from one state of the parser to another while graph-based algorithms operate on the set of all possible complete parse trees of the sentence in the search for the highest scoring one. Therefore the crucial difference between the two frameworks is whether the parsing model assigns scores to the transitions or to the dependency graphs.

McDonald and Nivre (2007) carried out an error analysis that showed that transition-based and graph-based systems produced different types of errors, which can be attributed to their theoretical properties, and built a combined system (Nivre and McDonald, 2008a). Below, we will discuss the Malt (Nivre et al., 2007b) and MST (McDonald et al., 2005b) parsers representing transition- and grammar-based approaches correspondingly and the Bohnet and Nivre (2012) parser, as an example of a combined model, which was the state-of-the art dependency parser in 2012-2013.

Malt MaltParser (Nivre et al., 2007b), dubbed Malt in the following, is a transition-based dependency parser with *local learning* and *greedy search*. This parser exploits *discriminative machine learning* with *history-based feature models* and supports several *deterministic parsing algorithms*.

A *transition* is a partial function that maps non-terminal configurations to new configurations. The *configuration* of the transition system is a triple consisting of a stack of partially processed nodes, a buffer of remaining input nodes, and a set of labeled arcs.

Malt is trained to approximate a globally optimal solution by making a series of *locally* optimal decisions. The system scores transitions between parser configurations based on the parse history and then *greedily searches* for the highest-scoring transition sequence that derives a complete dependency graph (Hall, 2008).

In order to approximate an oracle, a classifier that chooses a single transition from the space of all possible transitions at each parsing step, the Malt parser implements *discriminative learning* methods such as support vector machines. The learning is local since the parameters are optimized on individual transitions, not series of transitions.

Malt uses complex *history-based feature models* that combine static and dynamic attributes of tokens in the parser history (Black et al., 1992; Magerman, 1995). Static properties of attributes, such as word form and part-of-speech tag assigned at the preprocessing step, receive a value before the parsing begins and remain unchanged during parsing of the sentence, while dynamic properties, such as dependency type, are instantiated during parsing. Reference to the attributes of arbitrary tokens from the parser history is realized via address functions such as the i th token from the top of the stack, the i th token in the buffer, the head or sibling of token i in the partially built dependency graph, the leftmost or the rightmost child of token i in the partially built dependency graph.

Operation	Stack	Buffer	Partial dependency tree
Initialization	[ROOT]	[Salmon is a nutritious food]	ROOT Salmon is a nutritious food
Shift	[ROOT Salmon]	[is a nutritious food]	ROOT Salmon is a nutritious food
Left-Arc _{sbj}	[ROOT]	[is a nutritious food]	ROOT Salmon is a nutritious food
Shift	[ROOT is]	[a nutritious food]	ROOT Salmon is a nutritious food
Shift	[ROOT is a]	[nutritious food]	ROOT Salmon is a nutritious food
Shift	[ROOT is a nutritious]	[food]	ROOT Salmon is a nutritious food
Left-Arc _{amod}	[ROOT is a]	[food]	ROOT Salmon is a nutritious food
Left-Arc _{det}	[ROOT is]	[food]	ROOT Salmon is a nutritious food
Right-Arc _{obj}	[ROOT is food]	[]	ROOT Salmon is a nutritious food
Termination	[ROOT is]	[]	ROOT Salmon is a nutritious food

Figure 2.7: Syntactic analysis of the sentence “Salmon is a nutritious food” with the arc-eager algorithm

Most parsing systems prior to the Malt parser (Collins, 1997, 1999; Charniak, 2000) relied on non-deterministic algorithms with underlying generative models. The n-best list of output analyses of such systems can be re-ranked with discriminative-based algorithms that use the feature sets transcending the ones exploited by the original parsing models (Charniak and Johnson, 2005). In contrast, the Malt parser implements a *deterministic* parsing algorithm that makes locally optimal decisions guided by the classification model.

The MaltParser 1.7.2 package provides three families of parsing algorithms: Nivre (Nivre, 2003), Covington (Covington, 2001) and Stack (Nivre, 2009). Nivre’s algorithm runs in linear time and is limited to projective dependency structures. Covington is a quadratic-time algorithm that can be run in a mode that restricts the output to projective dependencies and in a mode that can produce non-projective dependencies. The Stack algorithm is similar to Nivre’s algorithm, but it has modes that allow the parser to produce non-projective trees. The algorithms for projective parsing can be combined with pseudo-projective pre- and post-processing (Nivre and Nilsson, 2005) in order to yield non-projective trees.

We will illustrate the parsing process on an example sentence using the arc-eager algorithm (Nivre et al., 2007b) which is a variant of the shift-reduce algorithm. An abstract machinery is composed of two data structures: a stack and a buffer. There are four types of transitions in

the system: Left-Arc, Right-Arc, Reduce and Shift. The transition Left-Arc adds an arc with dependency label from the top node in the buffer to the last node in the stack and pops the stack. The transition Right-Arc makes the top token in the buffer right-dependent of the last element of the stack and pushes the top token of the buffer to the stack. The transition Reduce pops the stack. The transition Shift pushes the top token in the buffer to the stack. The parser is initialized with an empty stack and a buffer containing all the words of a sentence. In the terminal configuration the buffer is empty.

Figure 2.7 shows the parsing process of the sentence “Salmon is a nutritious food” with the arc-eager algorithm.

Some advantages of the parsing techniques implemented in Malt are parsing time, which is linear in the size of the input on average (and quadratic in the worst case), the sufficiency of small amounts of training data and cross-language portability (Nivre et al., 2007b). In addition the algorithms are not restricted in the number of graph arcs in the feature representation. One of the main drawbacks of the deterministic parsing approach is error propagation.

Malt has better performance on shorter sentences which allow it to exploit rich feature sets with a reduced chance of error propagation (McDonald and Nivre, 2007). The Malt parser is more accurate for shorter dependency arcs and dependency arcs located further away from the root since these types of dependencies are constructed at early stages of the parsing process which leads to lower likelihood of error propagation. Malt is prone to over-predicting root modifiers because at the end of the parsing procedure all unattached tokens are linked to the root (McDonald and Nivre, 2007). Malt has a tendency to select earlier appearing verbs as heads of later appearing verbs (Goldberg and Elhadad, 2010).

MST MST (McDonald et al., 2005b) is a graph-based dependency parser with global near-exhaustive search.

MST tackles the problem of constructing a dependency tree as the search for a maximum spanning tree in a directed graph. A maximum spanning tree (MST) of a graph with weighted edges is a tree built on a subset of the edges from the graph that maximizes the sum of the edge scores such that every vertex in the graph appears in the tree. The maximum projective spanning tree of a graph is constructed similarly except that it can only contain projective edges relative to some total order on the vertices of the graph.

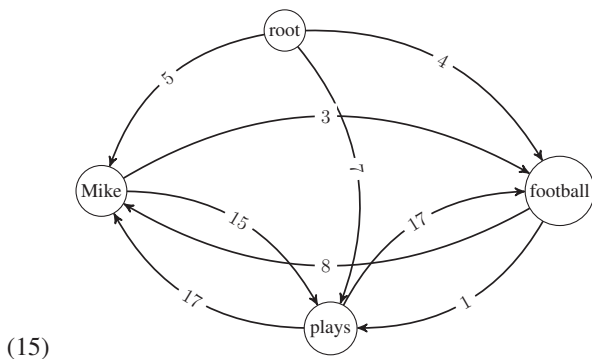
MST implements the online learning structural algorithm for large-margin multi-class classification problems called Margin Infused Relaxed Algorithm (MIRA) (McDonald et al., 2005a). Online learning iterates over training instances (pairs of a sentence and its correct dependency tree) updating the parameter vector by application of an update rule to the instance in question. Dependency parsing can be seen as a large-margin multi-class classification problem with the k highest scoring incorrect dependency trees as classes for a given sentence. The update rule optimizes the parameters of the model to maximize the difference between the scores of correct and incorrect dependency trees for sentences in a training corpus.

Owing to tractability constraints, the learning algorithm restricts the number of graph arcs in the feature representation. MST exploits three classes of features: (i) features over the parent-child node pairs, (ii) features that consider words occurring between a child node and its parent, (iii) features that look at words before and after the parent-child node pairs.

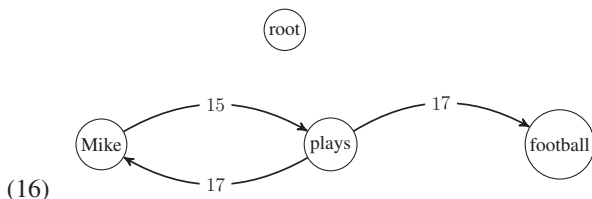
MST realizes projective dependency parsing with Eisner’s dynamic programming algorithm

(Eisner, 1996) and non-projective dependency parsing with Chu-Liu-Edmonds maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967). Unlike the pseudo-projective parsing algorithms of the Malt parser described in (Nivre and Nilsson, 2005), the Chu-Liu-Edmonds algorithm implements actual non-projective dependency parsing. The complexity of the Chu-Liu-Edmonds algorithm for non-projective dependency parsing is quadratic in the size of the input sentence. The Eisner algorithm has cubic complexity in the size of the input since it has to additionally enforce the non-crossing constraint of projective trees.

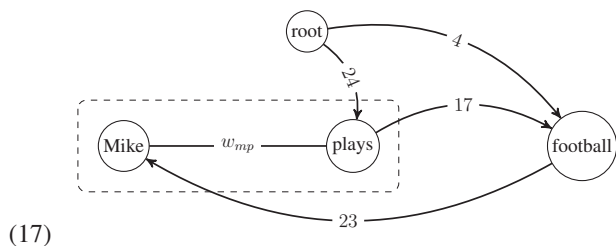
Below we describe the application of the Chu-Liu-Edmonds algorithm to an example sentence following (McDonald et al., 2005b). Initially the example sentence “Mike plays football” is represented as a directed weighted graph with all tokens attached to the root and interconnected between each other, see Example (15).



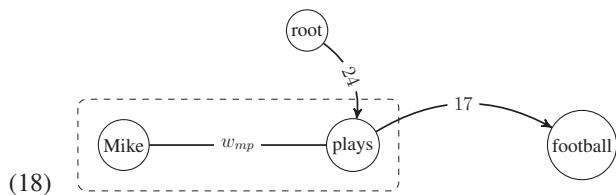
For each token of the sentence the algorithm identifies the incoming edge with the maximum weight, see Example (16).



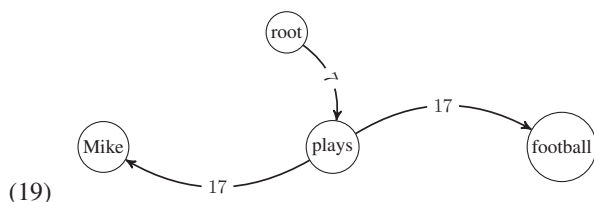
If the resulting structure is a tree, it would be the maximum spanning tree. In our running example the result is not a tree. The cycle should be contracted into a single node w_{mp} and the weights are recalculated as shown in Example (17). The edge from $root$ to w_{mp} is assigned a new weight of the best spanning tree including only the $root$ node and the vertices from w_{mp} : the score 24 is a sum of the original weight 7 of the edge from $root$ to $plays$ and the weight 17 of the edge from $plays$ to $Mike$. The edge from $football$ to w_{mp} is also assigned a weight of the best spanning tree rooted at the node $football$ and including the vertices from w_{mp} : the score 23 is a sum of the original weight 8 of the edge from $football$ to $Mike$ and the weight 15 of the edge from $Mike$ to $plays$.



The key property of the Chu-Liu-Edmonds algorithm is that an MST in the contracted graph can be transformed into an MST in the original graph. The algorithm recursively performs a greedy search of the best incoming edge to all tokens, as illustrated in Example (18).



Since the resulting structure in our example is a tree, it is the MST which allows us to reconstruct the MST of the original graph, as demonstrated in Example (19).



According to the comparative analysis carried out in McDonald and Nivre (2007), MST is more precise than Malt for longer dependency arcs, for arcs closer to the root, for arcs with more siblings and for arcs with a higher degree of non-projectivity, and the errors of MST are distributed over the graph more evenly than the errors of Malt. The decreased accuracy of Malt is due to error propagation and the attachment of the disconnected nodes of the graph to the root node at the end of the parsing. The two parsers furthermore show different accuracies over parts of speech: Malt is more accurate for nouns and pronouns whereas MST performs better for all other categories, especially conjunctions. McDonald and Nivre (2007) explain these observations by the fact that nouns and pronouns are typically attached further away from the root while verbs, adverbs and conjunctions are usually attached closer to the root. MST fails to analyze correctly coordination of NPs containing PPs which might be due to the feature PoS tags of tokens between head and dependent tokens used in MST (Goldberg and Elhadad, 2010).

Bohnet and Nivre (2012) parser The Bohnet and Nivre (2012) parser, dubbed B&N in the following, is a system for *joint part-of-speech tagging and non-projective labeled dependency parsing*. It is based on the *transitional model with beam search* and *global structural learning* employing rich *non-local feature representations*.

The B&N parser combines the best properties of transition-based and graph-based dependency parsing. In the spirit of the former strategy it implements a deterministic, highly efficient

transition-based model which exploits rich feature representations and in the spirit of the latter approach it relies on global learning and non-greedy search.

Unlike the best parsers based on PCFG models, most dependency parsers prior to B&N did not offer an option to perform part-of-speech tagging before or during parsing but required the input to be pre-processed with lexical categories. The experimental results in Bohnet and Nivre (2012) show superior tagging and parsing accuracy for morphologically rich languages like Czech and German and more configurational languages like Chinese and English by joint tagging compared to the pipeline systems that are more prone to error propagation.

We discussed the transition system and its components in Section 2.2.1. B&N adopts and extends the transition system of Nivre (2009) exploiting the following set of transitions: Left-Arc, Right-Arc, Shift and Swap that allow parsing of arbitrary non-projective trees. Like in the arc-eager algorithm in the Malt parser, transitions operate on two data structures: stack and buffer, and the Left-Arc and Right-Arc create a labeled dependency arc between the two nodes on top of the stack and replace the nodes with the head of the arc (rightmost and leftmost node correspondingly). The Swap transition moves the second topmost node from the stack back to the buffer. The Shift transition pushes the first node in the buffer onto the stack and labels it with the part-of-speech tag. Every word has to be pushed onto the stack in a Shift transition before termination of the parsing procedure to ensure that every word of the output dependency tree is assigned a lexical category.

Unlike Malt that relies on greedy best-first inference, the B&N parser is based on the beam-search algorithm. A beam is a list that stores weighted hypotheses and gets updated at every iteration of the search algorithm. The parameters of the inference algorithm determine the number of hypotheses allowed in the beam and define the constraints on transitions limiting the search space.

The learning algorithm implemented in B&N is a variant of the structured perceptron (Collins, 2002) in combination with beam search. This is a global learning method that tries to maximize the accuracy over the full sentence and not on individual local transitions. Zhang and Nivre (2012) showed that the combination of global learning and beam-search reduces error propagation and enables more powerful parsing models without overfitting.

The parser exploits rich non-local feature representations that refer to different aspects of parser configurations: baseline parser features, specialized tagging features, graph-based completion features and cluster features. The first group of features, baseline parser features adapted from Zhang and Nivre (2011), includes n-gram combinations of elements of the stack, buffer, arc set and candidate part-of-speech tags with corresponding scores. The group of specialized tagging features involves the i th best tag assigned to the first word of the buffer in combination with neighboring words, word prefixes, word suffixes, score differences and tag rank. Graph features are defined over the factors of a graph-based dependency parser as described in Bohnet and Kuhn (2012). Cluster features first used in Koo et al. (2008) are defined over word clusters generated with the Brown clustering algorithm (Brown et al., 1992).

2.2.2 Deep grammar parsing

Grammar-based parsers perform syntactic analysis of text relying on linguistically motivated grammars. In the following sections we will describe the ERG which is a hand-crafted

(1) ahead_of := (2) p_np_i_le &

[ORTH (3) < "ahead", "of" >]
	SYNSEM [LKEYS.KEYREL.PRED (4) _ahead+of_p_rel PHON.ONSET (5) voc]	

Figure 2.8: Lexical entry “ahead of” in the ERG lexicon

HPSG implementation, MRS, a framework for computational semantics used with the ERG, EDS, an MRS reduction into a variable-free dependency graph, and the PET parser that is often used for parsing with the ERG.

LinGO English Resource Grammar

The *LinGO English Resource Grammar (ERG)* (Flickinger, 2000) is a broad-coverage HPSG-based grammar of English, which has been developed by the DELPH-IN community effort since the early 1990s. DELPH-IN is an international partnership that combines linguistic knowledge and statistical algorithms in order to automatically analyze text meaning.

The ERG is designed in the grammar and lexicon development environment LKB for use with unification-based linguistic formalisms. The grammar is bidirectional, e.g. suitable for both parsing and generation. There are currently roughly 1000 lexical types and some 300 rules in the grammar. The early development and first application of the ERG was in the Verbmobil spoken language machine translation project (Wahlster, 2000). The grammar was originally engineered based on corpus data in informal genres such as conversations about scheduling and email regarding e-commerce. The ERG was later used for generation from semantic representations into English sentences in the Norwegian–English machine translation project called LOGON (Lønning et al., 2004) and in the course of the project, the grammar lexicon was enriched with vocabulary from the collection of English translations of Norwegian hiking texts from the LOGON corpus (Oepen et al., 2004). The ERG was employed for parsing the English Wikipedia (Read et al., 2012) as part of the WeSearch project, and the newspaper text of the Wall Street Journal within the framework of the DeepBank project (Flickinger et al., 2012). The grammar has a practical application in educational software for teaching writing in the on-line course of the Education Program for Gifted Youth (EPGY) at Stanford University (Suppes et al., 2012). The grammar has also been employed in intelligent auto-response software for commercial use.

Rules and lexical entries of the ERG are introduced in the TDL (Krieger and Schäfer, 1994) declarative language for typed feature structures specification. The ERG incorporates most of its linguistic knowledge in the lexicon entries. The core ERG lexicon was manually built, but it can also be enriched by automatically derived entries from various sources. Figure 2.8 shows a lexical entry “ahead of” from the ERG lexicon which consists of: (1) a lexical identifier, (2) a lexical type, (3) a stem, (4) a semantic predicate and (5) phonetic information.

The lexical identifier acts as the key for the lexical database entry. The lexical types of ERG, “word classes”, have the following structure: P_V_A_le, where

- “P” stands for a broad part-of-speech class (verb, noun, adjective, adverb, preposition, prepositional phrase, determiner, conjunction, complementizer, punctuation, miscellaneous);
- “V” is a valency field that contains the ordered sequence of complements for the given type;
- “A” means annotations, a fine-grained distinction among types with the same part-of-speech and complement. Annotations encompass category-internal subdivisions, e.g. mass vs. count vs. proper nouns, intersective vs. scopal adverbs, referential vs. expletive pronouns;
- “le” which stands for “lexical entity” is an invariant suffix used to facilitate regular-expression searches within the grammar source files.

In our example from Figure 2.8 `p_np_i_le` is an intersective (i) preposition (p) that takes a noun phrase (np) as a complement. Lexical types are arranged in a hierarchy which uses inheritance to avoid redundancy.

The grammar rules with the two main classes of lexical and construction rules are arranged into a hierarchy to support sharing of specifications. Lexical rules are divided into classes of inflectional and derivational rules that handle morphological phenomena and punctuation rules that assign punctuation as word affixes. The construction rules reflect HPSG constructions some of which have been introduced in Section 2.1.3, e.g. Head-Modifier, Head-Complement, Head-Specifier.

The syntactic level of analysis may be expressed in the form of a derivation tree that records how an analysis for an input sentence is derived (see Figure 3.3). This output format represents the typed feature structure of the sentence in a compact format. The derivation tree is a context-free phrase structure tree composed of leaf nodes corresponding to lexical entries (“child”), and intermediate nodes corresponding to grammar rules (`n_sg_ilr`—singular noun inflectional rule). Such a tree summarizes the results of a parse which is useful for grammar checking and ensuring that revisions to the parser do not have serious flaws (Copestake and Flickinger, 2000). The full HPSG analysis (typed feature structure) of a sentence can be reconstructed deterministically, given the grammar and the derivation tree with the unique identifiers of the lexical entries. As the full derivation depicts the feature structure, the sub-trees do so as well, and each such structure contains hundreds of feature-value pairs.

The derivation tree in Figure 3.3 represents HPSG concepts that were introduced in Section 2.1.3: HPSG constructions such as subject-head (`sb-hd_mc_c`), specifier-head (`sp-hd_n_c`), head-complement (`hd-comp_u_c`), and HPSG lexical rules such as orthography-invariant inflectional rule (`n_sg_ilr`) and orthography-changing inflectional rule (`v_3s-fin_olr`).

MRS

The ERG uses *Minimal Recursion Semantics (MRS)* (Copestake et al., 2005) as the meaning representation layer. MRS is a framework for computational semantics that uses “flat”

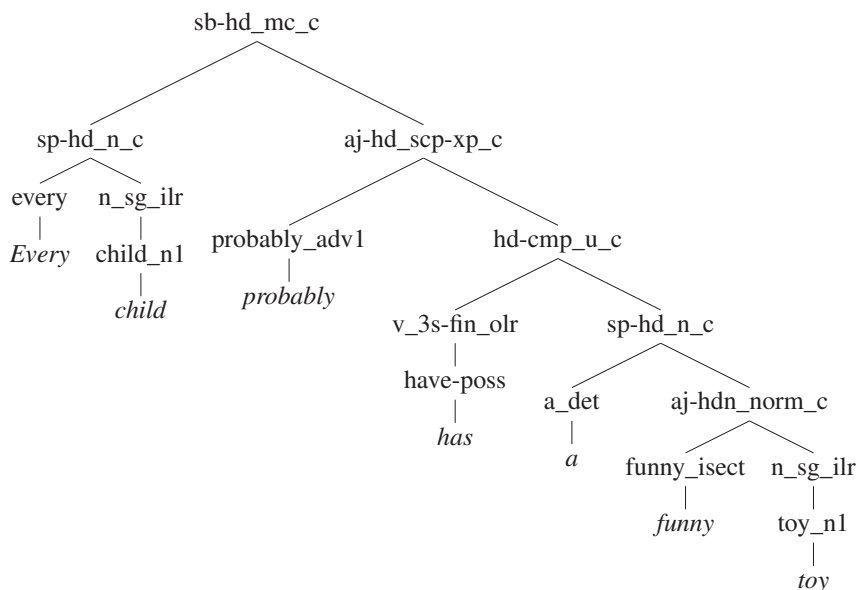


Figure 2.9: Derivation tree for the sentence “Every child probably has a funny toy”.

(non-embedded) structures. MRS can be used with a large-scale grammar for parsing, generation and semantic transfer and it can be implemented within a typed feature structure logic. The four main criteria that the MRS meta-language seeks to achieve are expressive adequacy, grammatical compatibility, computational tractability and underspecifiability. The principle of *expressive adequacy* requires the framework to allow linguistic meanings to be expressed correctly. *Grammatical compatibility* guarantees that semantic representations are linked “cleanly” to other kinds of grammatical information such as syntax (Copestake et al., 2005). *Computational tractability* means that it is possible to process meanings and to check semantic equivalence efficiently. *Underspecification* is the deliberate omission of information from linguistic descriptions which allows to encapsulate several alternative realizations of some linguistic phenomenon in a single representation. The motivation to use underspecified semantic representations is to avoid resolving all the ambiguities in a sentence since such resolution may require contextual and world knowledge, and may be computationally intractable.

The primitive structures of MRS are *elementary predications* (EP), similar to predicates in first-order logic, each expressing a single relation often corresponding to a single lexeme. The framework is called minimally recursive because EPs cannot be embedded into each other. Example (20) demonstrates an EP.

(20) $h_4: _every_q\langle 0:5 \rangle (\text{ARG0 } x_6 \{ \text{PERS } \beta, \text{NUM } sg, \text{IND } + \}, \text{RSTR } h_7, \text{BODY } h_5)$

An EP is a tuple of four components:

- a handle which is the label of the EP (h_4);

h	a handle or a label
x	an instance variable, introduced by a noun or an adjective
e	an event(uality) variable, introduced by a verb or an adverb
i	(for individual) an underspecification for x or e
u	(for unspecific or maybe unbound) an underspecification for x, e or h

Table 2.1: Types of variables in MRS

- a name of the relation (*_every_q_rel*);
- a list of zero or more ordinary variable arguments of the relation (nominal variable introduced by a noun/adjective (*x6*), third person (*PERS 3*), singular (*NUM SG*), count (*IND +*));
- a list of zero or more handles corresponding to scopal arguments of the relation (*h7* and *h5*).

Handles identify EPs being used for labelling purposes and as arguments inside elementary predications. An EP conjunction is a bag of EPs that have the same label.

Names of relations that describe lexical words (*_every_q_rel*) start with an underscore followed by the lemma (*every*), part-of-speech (*q* for quantifier) and a type descriptor (*rel* for relation) separated by underscores. Optionally the relation names may include a sense identifier to disambiguate different meanings of words with the same lemma and part-of-speech (e.g. “view” as a visual percept of an area or as a personal belief). Names of abstract predicates introduced by constructions have similar structure but without a leading underscore (e.g. “compound_rel”).

Various types of variables that act as *arguments* of relations in the context of MRS are shown in Table 2.1. Arguments can be characterized by properties such as person, number and gender for nominal variables and tense and mood for event variables. Example in Figure 2.10 shows the three kinds of EP: non-scopal: *_child_n_1(x6)*, *_have_v_1(e3, x6, x12)*, *_funny_a_1(e17, x12)*, *_toy_n_1(x12)*; quantifier EPs: *_every_q(x6,h7,h5)*, *_a_q(x12,h15, h14)*; and fixed scopal EP *_probable_a_1(e9, h10)*; *h1, h4, h8, h2, h11, h13, h16* are labels and *h7, h5, h10, h15, h14* are holes and *x6, x12* are nominal variables and *e3, e9, e17* are event variables.

Scopal relations are defined by the RSTR (“restriction”) and BODY values of the quantifier. The handle value of RSTR establishes the link between the quantifier and its noun relation forming a noun phrase (“every child”, “a toy”). The value of BODY is a handle variable that defines a scope of the quantifier and is not connected to any EP in an underspecified MRS. Thus, the MRS shown in Figure 2.10 for the sentence “Every child probably has a funny toy” underspecifies whether the funny toy is the same or different for every child.

MRS representations have the following structure:

< top handle, EP bag , bag of handle constraints >

$$(21) \quad \left\langle \begin{array}{l} h_1, \\ h_4: _every_q(0:5)(ARG0 x_6, RSTR h_7, BODY h_5), \\ h_8: _child_n_1(6:11)(ARG0 x_6), \\ h_2: _probable_a_1(12:20)(ARG0 e_9, ARG1 h_{10}), \\ h_{11}: _have_v_1(21:24)(ARG0 e_3, ARG1 x_6, ARG2 x_{12}), \\ h_{13}: _a_q(25:26)(ARG0 x_{12}, RSTR h_{15}, BODY h_{14}), \\ h_{16}: _funny_a_1(27:32)(ARG0 e_{17}, ARG1 x_{12}), \\ h_{16}: _toy_n_1(33:36)(ARG0 x_{12}) \\ \{ h_{15} =_q h_{16}, h_{10} =_q h_{11}, h_7 =_q h_8, h_1 =_q h_2 \} \end{array} \right\rangle$$

Figure 2.10: MRS for the sentence “Every child probably has a funny toy”

Each scope-resolved MRS represents a possible linguistic reading of a sentence described by an MRS. Scopally resolved MRS representations are essentially trees with EP conjunctions as nodes and dominance is determined by the ordering of EPs according to their scopes. The *top handle* of the MRS corresponds to a handle of the highest EP conjunction in the scope-resolved MRS and allows, for example, embedding of a clause in a longer sentence.

The *bag of handle constraints* on scope relations is introduced using *qeq* relations (equal modulo quantifiers) which show connections between holes (gaps in the semantic structure) and labels. Scope-resolved MRSs are derived by equating the holes with EP labels and unlike MRS itself which does not allow embedded structures, scope-resolved MRSs correspond to logic formulae with embedded predicates.

In the ERG analyses, MRSs are derived as feature structures of type *mrs* with features (i) HOOK that includes LTOP (top handle) and INDEX, (ii) RELS that represents a bag of EPs and (iii) HCONS that stands for handle constraints. Co-indexation in a feature structure is used to express variable identity. An example of the full MRS for the sentence “Every child probably has a funny toy” is shown in Figure 2.11. HOOK is used to make some of the information of the MRS externally available for semantic composition, e.g. combining the given MRS with other MRSs. The value of the INDEX is usually the ARG0 event variable of the main verb (in our example INDEX would refer to the event variable e_3 which is the ARG0 of the relation $_have_v_1$).

Elementary Dependency Structures

Open and Lønning (2006) proposed *Elementary Dependency Structures (EDS)* which is a reduction from a full logical-form MRS into a variable-free semantic dependency graph. Each relation is prefixed with its distinguished identifier¹ which can be equated with ARG0 in ERG. The semantics of a verb introduces an eventuality therefore a distinguished identifier for a verbal predicate is an event identifier, whereas the semantics associated with a noun introduces an instance which is expressed with a referential index in a nominal elementary predicate:

$$(22) \quad \text{dogs bark: dog}(x), \text{ bark}(e, x)$$

¹Distinguished identifier is alternatively known as inherent, or main, or characteristic identifier.

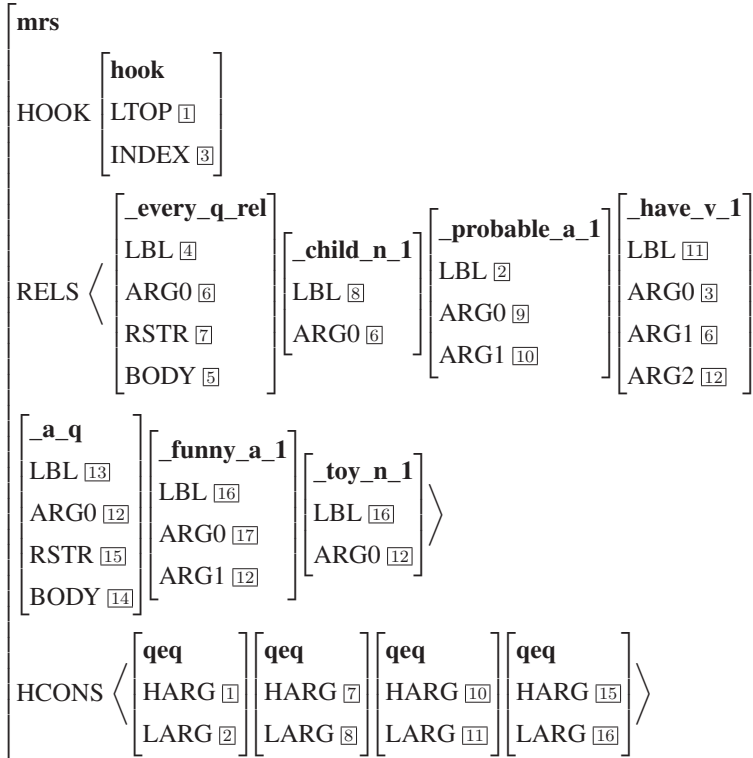


Figure 2.11: MRS for the sentence “Every child probably has a funny toy” in feature structure representation

```

e9 :
  _1 :  _every_q < 0 : 5 > [BV x6]
  x6 :  _child_n_1 < 6 : 11 > []
  e9 :  _probable_a_1 < 12 : 20 > [ARG1 e3]
  e3 :  _have_v_1 < 21 : 24 > [ARG1 x6, ARG2 x12]
  _2 :  _a_q < 25 : 26 > [BV x12]
  e17 :  _funny_a_1 < 27 : 32 > [ARG1 x12]
  x12 :  _toy_n_1 < 33 : 36 > []

```

Figure 2.12: EDS for the sentence “Every child probably has a funny toy”

Originally this representation was developed to simplify downstream processing of MRS in tasks like information extraction. Oepen and Lønning (2006) applied the representation to MRS banking, e.g. the process of manually selecting gold-standard semantic analyses from multiple analyses generated by the parser. EDS provides an interface that allows a human user to quickly compare semantic properties of multiple analyses of the same sentence. Fujita et al. (2007) used EDS to extract semantic features for parse selection and Dridan and Oepen (2011) introduced a parser evaluation metric, called Elementary Dependency Match (EDM), based on EDS.

The EDS representation of the MRS from Figure 2.10 is shown in Figure 2.12. EDS is a directed graph with semantic predicates (*_every_q*, *_child_n_1*) labeling its nodes, semantic argument roles (*ARG1*, *ARG2*) labeling its arcs and the first node as the root (*e₉*). Each node has its unique identifier (*_1*, *x₆*, *e₉*), characterization span (*< 0 : 5 >*) and a list of arguments (*[ARG1 x₆, ARG2 x₁₂]*). In our example the node *_probable_a_1* is connected to the node *_have_v_1* with the arc *ARG1*. Nodes can also take one or more constant arguments (e.g. *x₁₃ : named < 15 : 27 > (“Los_Angeles”)*).

We will return to EDS in Chapter 3 discussing the reduction from the semantic layer of the English Resource Grammar to bilexical dependencies.

PET parser

PET (Callmeier, 2000) is an open-source *agenda-driven bottom-up chart parser*. It is available as stand-alone software as well as an integrated module of the [incr tsdb()] profiling environment. Initially PET was developed for testing unification algorithms. Later on it was enhanced with *subsumption-based packing* (Oepen and Carroll, 2000), *selective unpacking* and *statistical parse ranking* (Carroll and Oepen, 2005; Zhang et al., 2007b) and *übertagging* (Dridan, 2013a) for increased efficiency.

PET builds the HPSG analyses in a *bottom-up* fashion from the input tokens up using an *agenda* to define the order in which chart entries are processed. The *chart parsing* algorithm (Kay, 1986) runs on a lattice of input tokens that are retrieved from raw text using the built-in preprocessor. Initially the raw text is tokenized following the PTB conventions with the framework REPP (Regular Expression-Based Pre-Processing), a set of ordered finite-state string rewriting rules (Dridan and Oepen, 2012). Part-of-speech tags are assigned with an efficient statistical tagger TnT (Trigrams’n’Tags) (Brants, 2000) and the tokenization style is adapted



Figure 2.13: Example of flat (on the left) and extended (on the right) analyses of the noun phrase “Air Force contract” from Vadas and Curran (2007)

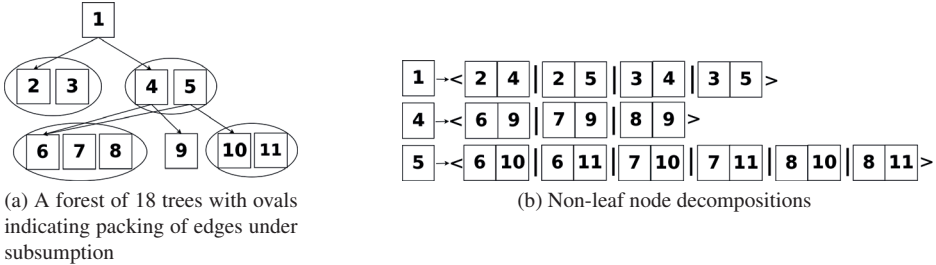


Figure 2.14: Packed forest with sub-node decompositions (similar to an example in Carroll and Oepen (2005))

to the expectations of the grammar with the chart-mapping rewrite formalism (Adolphs et al., 2008). In order to generate candidate lemmas, morphological rules are applied to each input token. Each suggested lemma is looked up in the lexicon with a subsequent check to see whether the applied morphological rule is applicable to the lexical item from the lexicon. Only valid lexical items are added to the chart and unified with other lexical items according to the grammar rules.

The number of possible analyses grows exponentially with the sentence length which results in exponential worst-time complexity if the parser attempts to build all the parse trees explicitly. *Ambiguity packing* relies on the observation that a given structure from one tree may be shared by many other trees in the parse forest, even though none of them are globally identical. An example of an ambiguity is the PP-attachment problem that we saw in Section 2.1.1 in the sentence “A man saw a star with a telescope” where prepositional phrase “with a telescope” can modify either “saw” or “a star”. Since all the lexical items and the prepositional phrase are the same in the possible analyses, it is important to avoid duplication of information in order to optimize memory usage. Such ambiguities are tackled by representing repeating elements of the analysis as single chart edges.

Another example of an ambiguity is an analysis of the internal structure of the noun phrase “Air Force contract” shown in Figure 2.13. Such ambiguities lead to an exponential number of parallel derivations therefore they are packed into a single object to avoid creating a separate construct for each alternative. Figure 2.14a (similar to an example in Carroll and Oepen (2005)) illustrates a packed forest comprising 18 trees with the nodes 2 and 3 that could correspond

to the analyses from Figure 2.13 for example. The local packing is based on the subsumption relation (discussed in Section 2.1.3) instead of equivalence which is common for packing in CFG parsing, because there is a very small probability of two feature structures being exactly equivalent.

The *selective unpacking* algorithm developed by Carroll and Oepen (2005) and further extended by Zhang et al. (2007b), applies a conditional Maximum Entropy model to find the n-best analyses from the packed forest. Carroll and Oepen (2005) introduce two concepts that are important for the forest unpacking: i) *decomposing* edges locally (see Figure 2.14b) and ii) nested contexts of “horizontal” search for ranked *hypotheses* (i.e., uninstantiated edges) about candidate sub-trees. The basic algorithm is a graph search through the forest that tries to postpone the most expensive unification operations and perform them only on the best hypotheses.

Another statistical method implemented in the PET parser, called *übertagging* (Dridan, 2013a), exploits lexical information for improved parsing. Übertagging is an extended variant of supertagging (Bangalore and Joshi, 1999), a technique that attempts to localize the computation of linguistic information by linking lexical items with rich descriptions (supertags) that enforce constraints in a local context. Dridan (2009) investigated optimal granularity of tag forms and found that applying supertagging to parsing with the ERG is challenged by two factors:

- a supertag by definition is an atomic structure containing all the information about a lexical item while in the ERG this information is not centralized in the lexicon, but comes from different sources (e.g. lexical rules denote inflectional information);
- multiword expressions like “all of a sudden” introduce segmentation ambiguity between managing them as single units or processing component tokens individually.

Übertagging determines segmentation and supertags simultaneously and thus avoids heuristics for approximation of the ambiguous tokenization.

2.3 Parser combination

Parser combination is a general term for various methods of combining parsers in order to achieve improved parsing performance. The aim of parser combination is producing results that are superior to those of each of the individual parsers. One of the prerequisites for advantages of ensemble systems is complementarity of the errors that different parsers make. For example, the error analysis of McDonald and Nivre (2007) showed that transition- and graph-based systems are good candidates for combination due to this reason. Combination of “deep” and “shallow” analyzers is usually motivated by the potential to achieve a good balance between accuracy, robustness and efficiency. In the following subsections we will introduce various approaches to parser combination for data-driven and grammar-based parsing.

Enabling a data-driven parser to learn from other parsers

Parse hybridization combines the substructures of several parses. Henderson and Brill (1999) proposed two variants of a constituent parse hybridization, that ensure no crossing brackets in the output tree: constituent *voting* which includes a constituent in the parse tree if enough

parsers agree that it belongs in the parse, and *training a classifier* to determine which constituents to add to the parse tree. Empirical evaluation showed that both methods helped to improve robustness and reduce precision and recall error rates.

Zeman and Žabokrtský (2005) applied the ideas of Henderson and Brill (1999) to dependency parsing. The problem with the above two strategies of the parse hybridization method is that they do not guarantee that the output will be a well-formed dependency tree.

Sagae and Lavie (2006) suggested parser hybridization via *rearsing*. Each sentence is analyzed with k different parsers and then a weighted parse chart is constructed from the constituents of the decomposed parses. The resulting parse tree can be built by determining the back-pointers indicating what smaller constituents the elements of the chart contain and this can be solved with a bottom-up chart parsing algorithm.

In application to dependency parsing, instead of a chart, a weighted graph is built with words of the sentence as nodes and dependencies obtained from the parses as directed edges. The optimal dependency structure is the maximum spanning tree for the graph and it can be found by reparsing the sentence using a dependency parsing algorithm. Sagae and Tsujii (2007) showed the utility of the reparsing approach also for the case when the variants of a single algorithm are combined.

Nilsson (2009) discussed *transition level* parser hybridization for transition-based dependency parsing predicting the next transition with a group of classifiers and choosing a single transition from the available options based on voting or machine learning.

Parser switching chooses which parser should be trusted for a particular sentence. Henderson and Brill (1999) described two alternatives for parser switching: *similarity switching* which selects the parse most similar to the other parses and *training a classifier* to determine the most probable solution. Zhang et al. (2008) implemented the latter method to combine transition-based and graph-based parsers in their syntactic and semantic dependency parsing system achieving high accuracy.

Zeman and Žabokrtský (2005) proposed a blended approach of parser hybridization and parser switching for dependency analysis that guarantees an acyclic output structure. At each step, all dependencies that introduce a cycle are dropped and if there are no valid dependencies for a word, the whole partial structure is discarded and the best analysis built by one of the component parsers is adopted instead.

In **parser stacking** the output of one parser supplies features for the other parser. Zhang and Clark (2008) were the first to implement a parser that integrated transition-based and graph-based models in one system by using global linear learning based on averaged perceptrons, the union of feature templates and the beam-search decoder from the transition-based parser. The combined parser showed superior accuracy levels on the English and Chinese Penn Treebank corpora over the systems representing each of the approaches.

Nivre and McDonald (2008b) carried out feature-based integration of the Malt and MST parsers extending the feature vector of MST with a certain number of features generated by Malt and training a classifier to determine in which situations to rely on the additional features.

Similar to Nivre and McDonald (2008b), Zhang and Wang (2009) investigated the method of building feature-based models for statistical dependency parsers using a dependency backbone extracted from the ERG parser output. The native feature model of the MST parser was extended with features describing the uni-gram, bi-gram, PoS of words in between and PoS of

words surrounding a token in the dependency tree derived from parses from HPSG syntactic analyzer. For Malt additional features extracted from the HPSG dependency backbone concerned the top token in the stack and the head token of the buffer. The enhanced feature model led to accuracy drops on in-domain data but it helped to improve performance in terms of labeled and unlabeled attachment scores on out-of-domain test sets.

Analogously, Øvrelid et al. (2009) implemented a parser stacking approach to the combination of the LFG grammar-based parser and Malt. The procedure included conversion of LFG f-structures to dependency graphs, extension of the treebank with information from these graphs and modification of the Malt parser's feature model to refer to the additional features. This method proved to have positive impact on the accuracy of the dependency parser.

Reranking. Many parsing systems apply a two-stage approach to sentence processing: at the first stage, the parser has to produce an n -best list of analyses and at the second stage the task is to rescore the set of n -best parses in order to select the best parse, e.g. to rerank this set. In parser combination via reranking the outputs of multiple parsers can be jointly reranked or one parser can assist the reranking of the other.

Zhang et al. (2009a) introduced an approach for parser combination that utilizes tree probability from each individual parser and relies on n -best outputs of each parser instead of just 1-best results. First, n -best outputs of each of the k different parsers are combined and m unique analyses from $n * k$ trees are re-evaluated with the k models which are used by the k parsers. In addition, some feature scores for each tree are computed. The overall score for each tree is estimated by a linear function that balances the model and the feature weights. Finally, the m trees are reranked and the one with the highest score is selected as an output.

Farkas et al. (2011) exploited features extracted from analyses of the Bohnet (2010) dependency parser for reranking of the PCFG parser for German called BitPar (Schmid, 2004). By incorporating information from one formalism for reranking another authors achieved improved accuracy.

Ren et al. (2013) combined constituent and dependency parsers via a factored model so that a dependency parser reranked the k -best outputs from a constituent parser. The two parsers complement each other: the constituency parser reduces the ambiguities generated by the dependency parser while the latter resolves long distance dependencies between lexical items.

Restricting the search space of a grammar-based parser by a data-driven parser

In contrast to the approaches discussed above, parser combination methods for grammar-driven parsing usually aim to improve efficiency and robustness rather than accuracy. Large-scale grammars often introduce a high degree of ambiguity that leads to efficiency issues of deep natural language processing. For example, for comprehensive linguistic formalisms, such as HPSG, the issue of inefficiency caused by complicated data structures is a serious obstacle to their application to practical large-scale systems for natural language processing. A data-driven parser can pre-structure the search space of the deep parser within a hybrid system and improve the processing time. In the case that a deep parser fails to return a full analysis of the sentence, a data-driven parser can guide the selection of partial analyses from the chart and improve robustness.

CFG filtering techniques (Harbusch, 1990; Becker and Poller, 1998; Kiefer and Krieger, 2000; Torisawa et al., 2000) were first applied to Tree-Adjoining Grammars (Joshi et al., 1975)

and later ported to the HPSG framework. These techniques were proposed as an instrument to prune the hypothesis space of grammar-based parsers. Such methods eliminate partial parse trees that do not contribute to the final output using a CFG extracted from a given grammar prior to parsing. The main idea is to parse a sentence with a CFG first and then to run a grammar-based parser deterministically employing the derivations licensed by the CFG. The CFG parsing tends to be much faster than the grammar-based parsing because the combination of symbols that appear in a context-free rule is restricted, therefore an approach of guiding a HPSG parser with a CFG-forest leads to a speed-up. Matsuzaki et al. (2007) fully integrated CFG filtering with the supertagging-based HPSG parser for English using Kiefer and Krieger’s algorithm (Kiefer and Krieger, 2000) and obtained substantial improvements in efficiency.

PCFG filtering. Kiefer et al. (2002) proposed to use a PCFG for HPSG approximation in order to establish the ranking of CFG trees so that the grammar-based parser processed the most probable CFG trees first. Cahill et al. (2002) developed a methodology to automatically extract robust PCFG-based Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1982) approximations that allow parsing into trees and f-structure representations. Cahill et al. (2004) augmented the shallow grammars of Cahill et al. (2002) with long-distance dependency resolution in f-structures turning them into “deep” grammars. In a similar spirit, Zhang and Krieger (2011) experimented with a corpus-driven PCFG approximation of the English Resource Grammar showing potentials for improved robustness and efficiency though the method was not eventually integrated with the released versions of systems for parsing with the complete ERG.

Reranking. Similarly to Farkas et al. (2011), Kim et al. (2012) used the analyses of a dependency parser to generate features for reranking n-best outputs of the Combinatory Categorical Grammar (Steedman, 2000) parser called C&C (Clark and Curran, 2007b). Features were created by comparing Stanford-style dependencies derived from the C&C parser with dependencies generated by a dependency parser. The authors experimented with Malt and MST independently, training them on several dependency schemes: CoNLL 2007 shared task-style dependencies produced by the Penn2Malt utility, Stanford Basic dependencies from the Stanford parser, CoNLL dependencies from the software of Johansson and Nugues (2007) and dependencies obtained with the converter of Tratz and Hovy (2011).

Guiding a grammar-based parser with a dependency parser. Sagae et al. (2007) suggested a pipeline in which a dependency parser is applied before an HPSG parsing model. Surface dependencies constrain the application of wide-coverage HPSG rules which results in a significant improvement of accuracy. Firstly, a HPSG treebank is converted to CFG-style trees that are further transformed to dependency trees with the head-selection rules of Collins (1999); secondly, a statistical dependency parser is trained on the HPSG treebank in dependency representation; and, finally, the parsing algorithm is extended in such a way that the log-likelihood of partial parse trees created by HPSG schema applications is penalized in case the constraints of the dependency graph produced by the dependency parser are violated. During the HPSG parsing process information about the heads is available since it is encoded in the grammar, and therefore it is possible to verify whether the HPSG construction contradicts the restrictions posed by the dependency parser.

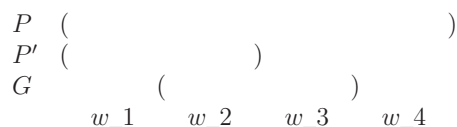


Figure 2.15: Crossing brackets. Bracket P is consistent with P' and G, while bracket P' crosses bracket G

2.4 Parser evaluation measures

General criteria for parser evaluation are efficiency, robustness and accuracy. *Efficiency* can be measured as the average total time and (peak) memory usage per utterance. *Robustness* is often expressed in terms of coverage: the percentage of sentences from a corpus that are assigned an analysis by a parser. *Accuracy* in the technical sense can be quantified in terms of whole-sentence precision: the proportion of correctly parsed sentences. *N-best accuracy* estimates for how many sentences the parse matching the gold standard is among the N best parses.

Accuracy in our definition measures so-called “exact match” which by many is considered problematic because it does not distinguish between poor quality and nearly correct analyses. For parser development it is important to have a deep understanding of which linguistic phenomena cause a system to fail, therefore metrics that allow one to break the analysis into component parts can be of great use. We provide a brief overview of some of these more granular parser evaluation methods in the following paragraphs.

PARSEVAL metrics

The PARSEVAL metrics (bracketed and labeled precision, bracketed and labeled recall, bracketed and labeled F-score, crossing brackets) were proposed by the Grammar Evaluation Interest Group (Harrison et al., 1991), extended by Grishman et al. (1992) and constitute the standard measures for evaluation of constituency parsers. The most commonly used software for evaluation with these metrics is the `evalb` toolkit developed by Satoshi Sekine and Michael Collins in 1997. *Precision* is defined as the percentage of correct constituents in the parser output. *Recall* is defined as the number of constituents from the gold standard that are present in the parser output divided by the total number of constituents in the gold standard. *F-score* is a weighted harmonic mean between precision and recall with β parameter which defines whether precision or recall have more importance for the system (with $\beta = 1$ recall and precision have equal significance):

$$F_{\beta} = \frac{(\beta^2 + 1) * \text{precision} * \text{recall}}{\beta^2 \text{precision} + \text{recall}}$$

Labeled measures check not only structural correctness, but also the correspondence of syntactic labels to a gold standard. Bracketed precision, recall and F-score are computed over just the bracket structure of the constituency tree, while labelled precision, recall and F-score evaluate

both brackets and labels. *Crossing brackets* is an average of the number of constituents from the parser output that cross over constituent boundaries of the gold standard where neither is properly contained in the other (see Figure 2.15).

PARSEVAL metrics have been criticized for their inability to reflect parser quality properly (Carroll et al., 1998; Briscoe et al., 2002). The weakness of the PARSEVAL metrics concerns overpenalization of some errors and incapability to detect others. Rehbein and Genabith (2007) showed that for German, the metrics favour annotation schemes with a high ratio of nodes per word and do not correlate well with other evaluation methods. Since PARSEVAL metrics are based on similarity estimation between phrase structure trees, they cannot be applied to grammars which produce analyses of other styles. Carroll et al. (1998) provide a detailed survey of other extant evaluation schemes.

Dependency-based evaluation metrics

The first dependency-based evaluation measures adopted by the computational linguistics community were precision and recall over WORD POS HEAD triples, as proposed by Lin (1998). *Precision* is the percentage of dependency relationships in the parser output that are also found in the gold triples. *Recall* is the percentage of dependency relations in the gold triples that are also found in the parser output triples.

GR-based evaluation (Carroll et al., 1998) is an approach to parser assessment which has the advantage of being language- and application-independent. *Grammatical relations (GRs)* describe the syntactic dependency which holds between a head and a dependent. The approach presupposes that a parser or a grammar is able to identify heads. For constituency parsers, heads can be identified with head-finding rules (Collins, 1999) while for dependency parsers and parsers based on linguistic theories such as LFG and HPSG heads are explicit in the analysis.

There are certain differences between dependency relations and grammatical relations:

- GRs are organized into a hierarchy and allow underspecification for parsers with incomplete knowledge of syntax;
- lexical items can have two heads in the GR analysis (for example for control relations as in “Paul intends to leave IBM” where “Paul” is headed by both “intends” and “leave”);
- GRs can express arguments that are not lexically realized.

Recall is the ratio of the number of GRs returned by the parser that match the GRs from the gold-standard annotation of the sentence, divided by the total number of GRs in the gold standard. *Precision* is the ratio of the number of GRs returned by the parser that match GRs from the gold-standard, divided by the total number of GRs returned by the parser for the sentence. In order to evaluate a syntactic analyser with GR recall and precision, GRs have to be read-off from sentences of a treebank and parser’s output.

Collins (1999) used a different dependency-based approach to evaluation calculating precision and recall over triples <MODIFIER, HEAD, RELATION>, where MODIFIER is the index of the dependent token, HEAD is the index of its lexical head and RELATION is a four-tuple of nonterminals <Parent, Head, Modifier, Direction> (e.g. <S, VP, NP-C, L> indicates what for English corresponds to a subject-verb dependency directed to the left).

The standard measures for dependency parsing evaluation currently are unlabeled attachment score (UAS) and labeled attachment score (LAS). The metrics are implemented in the standard evaluation software `eval` released for the CoNLL-X shared task 2006 (Buchholz and Marsi, 2006). The attachment score, or the *unlabeled attachment score (UAS)*, is computed as the proportion of tokens in a sentence (usually excluding punctuation) that are assigned the correct head (or no head if the token is a root). The overall attachment score is then calculated as the mean attachment score over all sentences in the sample. The *labeled attachment score (LAS)* is an attachment score where both the head and the label (dependency type) must be correct, but which is otherwise computed in the same way as the ordinary (unlabeled) attachment score. The *label accuracy score (LACC)* is the percentage of tokens with correct relation label, irrespective of the head. Additional measures useful for error analysis of particular graph properties discussed in McDonald and Nivre (2007) are the precision and recall relative to dependency length, dependency arc distance to the root, local neighborhood and degree of non-projectivity.

The CoNLL-X shared task evaluation script `eval.pl` evaluates the output of a parser with respect to a gold standard using UAS and LAS metrics and provides detailed information for each POS tag and for each dependency relation in terms of precision and recall.

Evaluation of grammar-based parsers

For grammar-based parsing, *coverage*, the percentage of utterances that a parser assigns an analysis, is an important metric during the grammar engineering process. For such grammars, like the ERG, coverage is still not always 100% when the accurate parsing mode with certain limits on memory and time is in use (Ivanova et al., 2013a). An *overgeneration metric*, the percentage of ungrammatical utterances that receive an analysis, can be useful when grammaticality judgements of the parser are analyzed.

Accuracy of computational grammars being developed within the DELPH-IN community is commonly measured as oracle and Top-1 precision. *Oracle precision* is the proportion of sentences for which a correct parse is among the analyses produced by the parser. This metric abstracts from the ranking model of the parser and evaluates only the grammar. *Top-1 precision* shows the proportion of sentences for which the correct parse is ranked the highest. It can be thus viewed as a method to evaluate the grammar and the parse ranking model as one system. *Mean reciprocal rank* is a metric that focuses solely on the quality of the parse ranker. Reciprocal rank is the reciprocal of the rank of the correct analysis, or zero if there is no parse matching the gold standard in the top- N analyses.

Dridan (2009) proposed *Elementary Dependency Match (EDM)* as a method for granular evaluation of the English Resource Grammar based on Elementary Dependency Structures (see Section 2.2.2 above) and their decomposition into basic semantic dependency triples. The metric is a computed over elementary dependencies (ED) of the ERG in the form:

- (a) relation_i ;
- (b) $\text{relation}_i \text{ role}_j \text{ relation}_k$;
- (c) $\text{relation}_i \text{ property}_j \text{ value}_j$.

Relations represent predicate names of elementary predicates (EPs); roles such as *ARG0*, *ARG1* refer to handles within EP; properties are attributes of MRS variables (e.g. PERS, NUM, TENSE); and values are instantiations for these properties (e.g. 3, *sg, past*). The character span is used instead of a relation name to avoid propagation of an error in the predicate name to each discriminant bearing that relation.

The method is similar to the evaluation approaches for CCG (Clark and Curran, 2007b) and (Miyao and Tsujii, 2008) for ENJU HPSG grammar-based parsers.

Cross-framework parser evaluation

One of the challenging questions for the parsing community is how to compare parsers based on different formalisms and producing different output, e.g. constituency parsers that generate phrase structure trees, dependency parsers that output dependency trees and grammar-based parsers that produce formalism-specific representations.

One of the approaches is to carry out training and testing of parsers on datasets from the same resource. The Penn Treebank originally designed in the framework of constituency grammar but also available in dependency and various grammar-based representations (Hockenmaier and Steedman, 2007; Flickinger et al., 2012), allows this form of cross-formalism comparison. However, the method has serious disadvantages: 1) parsers cannot be compared on other resources and 2) training and testing on the same data leads to overfitting (Clark and Curran, 2007a).

Another approach could be to convert the output of the parsers to some common representation for evaluation, such as grammatical relations or dependency relations. The complexity of conversion procedures for each parser plays a significant role because it establishes the upper bound on parser performance in the target representation (Clark and Curran, 2007a).

Rimell et al. (2009) propose *construction-based parser evaluation* using dependency representations. The approach implies the selection of fairly frequent linguistic phenomena, construction of gold standard analyses for them and evaluating whether parsers succeed in recovering the dependency structure correctly. Rimell et al. (2009) and Nivre et al. (2010) focus on long distance dependencies often called unbounded dependencies.

Bender et al. (2011) used the methodology for careful cross-framework comparison of seven parsers on 10 construction types, including both local and distant dependencies. This approach helps to overcome limitations of the over-optimistic PARSEVAL method due to the fact that PARSEVAL metrics do not take into account that some dependencies are “easier” (e.g. article-noun) and others are “harder” (e.g. preposition phrase attachment, coordination).

Extrinsic evaluation

The approaches to evaluation discussed above are called *intrinsic* and evaluate parsers as stand-alone systems. Such methodologies measure the performance of a parser in the framework it is developed by comparison of parser results to a gold standard. Intrinsic evaluation allows one to facilitate potential improvements in the development of a parser by identifying its weaknesses and also comparing different parsers to each other. In contrast, *extrinsic* evaluation estimates the impact of integrating a parser in some language technology application. This type

of evaluation measures the performance of the whole application. Extrinsic evaluation validates the benefits of embedding a parser in a natural language processing system.

Task-based parser evaluation has previously been performed, for example, in the work of Mollá and Hutchinson (2003) using the answer extraction system that operates over UNIX manual pages and in the work of Miyao et al. (2008) using the information extraction system that performs protein-protein interaction identification in biomedical papers. In the former publication a dependency parser output is used to construct logical forms whereas in the latter publication dependency paths extracted from a parser output are used as features for training a statistical classifier.

2.5 Linguistic resources

Natural language modelling requires annotated corpora and research efforts in different sub-tasks of the field have produced a variety of formats for corpus annotation. The annotated resources developed for natural language processing enable a wide range of research efforts in automatic syntactic analysis as well as in other fields. Researchers need access to large data collections in order to build realistic models of human languages. In the following, we will describe several resources that are used in our experiments.

Penn Treebank

The Penn Treebank (PTB) (Marcus et al., 1993) has become a central resource for statistical parsing of English. This treebank consists of a collection of hand-corrected annotations of American news text of the Wall Street Journal (WSJ) magazine and a portion of the Brown corpus (Kucera and Francis, 1967). The context-free grammar style bracketing annotation includes phrases, part-of-speech tags, null elements (such as those introduced by wh-movement and topicalization), function labels and other linguistic phenomena loosely based on the principles of the Government and Binding theory (Chomsky, 1981). The Penn Treebank is comprised of the phrase structure trees of the sentences represented in labelled bracketing format with the linguistic decisions documented in the annotation guidelines.

For the purpose of experimentation in the domain of bilexical dependencies, a number of converters have been implemented that perform a transformation from constituency to dependency trees, e.g. (Collins, 1999), (Yamada and Matsumoto, 2003), Penn2Malt (Nivre, 2006). One of the most prominent methods of such conversion from the Penn Treebank annotations to bilexical dependencies is introduced in Johansson and Nugues (2007). The resulting dependency scheme allows the expression of long-distance dependencies, non-projectivity, internal structure of some noun phrases and other linguistic phenomena encoded in the Penn Treebank. This scheme is more semantically oriented than its predecessors. Results in Elming et al. (2013) confirm that the Johansson and Nugues (2007) scheme is well-suited for down-stream applications such as negation resolution, semantic role labeling, statistical machine translation, sentence comprehension and perspective classification. The format was exploited in the CoNLL shared tasks on Dependency Parsing in 2007 (Nivre et al., 2007a), on Joint Parsing of Syntactic and Semantic Dependencies in 2008 (Surdeanu et al., 2008) and on Syntactic and Semantic

Dependencies in Multiple Languages in 2009 (Hajič et al., 2009). We will dub this scheme as “CoNLL” from here on.

Continuing the dependency tradition, de Marneffe and Manning (2008a) proposed the user-centered Stanford Dependency scheme in 2008 aiming to provide a simple application-oriented description of grammatical relations. The design of the Stanford representation was inspired by Lexical-Functional Grammar (Bresnan, 2001) which operates with a separate layer for grammatical functions (f-structure). The Stanford Dependency scheme departs from its theoretical roots in the goal to be a simple and practical model for relation extraction tasks (de Marneffe and Manning, 2008a). The main focus is on usability and intelligibility. There are two different realizations of Stanford dependencies that will be discussed in detail in Chapter 3: 1) “*basic*” in which each word of the sentence acts as a node of the dependency graph; 2) “*standard*” in which some words are collapsed into the labels of the dependency arcs. The latter representation is claimed to be closer to the semantics of the sentence (de Marneffe and Manning, 2008b).

The Stanford dependency format was applied to different tasks in the natural language processing domain, e.g. in PASCAL Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2006), in bioinformatics, e.g. for extraction of relations between genes and proteins (Fundel et al., 2007), in biomedical domain, e.g. for evaluation of parsers (Pyysalo et al., 2007), in opinion mining and sentiment analysis, e.g. for movie review mining (Zhuang et al., 2006). The Stanford dependency format has recently been extended to the Universal Stanford Dependencies scheme (de Marneffe et al., 2014) and Universal Dependencies (Nivre, 2015), a revised set of grammatical relations consistent across languages of different linguistic typologies.

DeepBank

DeepBank (Flickinger et al., 2012) is a corpus of HPSG analyses from the LinGO English Resource Grammar on top of the raw text of the Wall Street Journal part of the Penn Treebank built within the Deep Linguistic Processing with HPSG Initiative (DELPH-IN) infrastructure. DeepBank is created semi-automatically by parsing the data with the ERG parser and manual disambiguation using the discriminant-based approach. Version 1.0 covers sections 0-21 and lacks analyses for some 15% of the WSJ sentences, for which either the parser failed to build a forest of candidate analyses within the specified restrictions on time and memory usage, or the annotators found none of the available parses acceptable.

Redwoods

LinGO Redwoods (Oepen et al., 2004) is another treebank developed in the DELPH-IN environment in a fashion similar to the DeepBank treebank. The Redwoods treebank comprises ERG annotations of the data from a variety of genres and domains such as Verbmobil, e-commerce corpora, LOGON Norwegian-English MT corpus, English Wikipedia from WeScience treebank (Ytrestøl et al., 2009), Brown corpus (SemCor) and other sources with a total size of some 400,000 annotated tokens. Both DeepBank and Redwoods are implemented as dynamic treebanks with mechanisms to automatically update the resources with an enhanced version of the grammar. One of the main motivations of the Redwoods treebank development is to enable cross-domain parsing experiments since parsers trained on PTB exhibit large per-

formance drop on other domains (Gildea, 2001). Moreover, the language of PTB and the annotation decisions date back to the late 1980s.

2.6 Summary

In this chapter we first had a brief bird's-eye view of several syntactic theories underlying modern parsing technology, and then turned to a discussion of applied research topics: the somewhat diffuse borderline between data-driven and grammar-based paradigms, several state-of-the-art syntactic analyzers each representing a certain framework, a high-level overview of previous work on parser combination, present-day parser evaluation methods and their limitations, and the importance of annotated linguistic resources in the field with relevant examples.

Ongoing tendencies show a gradual shift of attention from representation of sentence analysis in the shape of phrase structure parse trees to bilexical dependencies. These trends are manifested by the development of direct dependency parsing software, diverse constituency-to-dependency conversion procedures, dependency treebanks, dependency representations and dependency-based evaluation measures. In the current thesis we explore such aspects of dependency parsing as differences and commonalities of contemporary dependency schemes, reduction of HPSG-based grammar formats to bilexical dependencies and the evaluation of parser performance with respect to different data formats and with respect to other kinds of software.

Community interest in domains other than the canonical text of Penn Treebank from the financial news realm, is continuously growing, and more and more effort is invested in exploration and annotation of user-generated content available on the Web. However, Penn Treebank still remains an important resource for parser comparison therefore we will present results both on the standard excerpt from PTB and on cross-domain sets of less formal data.

High-precision grammar-based parsers gained significant improvements in efficiency and coverage over the past decades via extensive research on development and integration of statistical processing techniques. Software based on hand-crafted grammars is mature and exhibits different strengths compared to the purely statistical parsing approaches. In this thesis we will carry out a parser comparison analysis that shows multiple dimensions of performance for representatives from constituency, dependency and HPSG frameworks.

In Chapter 3 we will work out a reduction from syntactic and semantic levels of the English Resource Grammar to bilexical dependencies and examine the relationship between the newly developed, HPSG-derived scheme to more standard ones, such as CoNLL Syntactic Dependencies and Stanford basic discussed above. Chapter 4 is dedicated to an empirical comparison of several dependency representations from Chapter 3 in parsing showing the difficulty for a statistical parser to process each of them. Chapter 5 presents experiments on head-to-head cross-framework parser comparison. Chapter 6 is on parser combination of the HPSG-based deep parser and statistical data-driven parsers.

Chapter 3

Syntactico-semantic dependencies

As we recall from the previous chapter, the PET parser, a software for parsing with DELPH-IN precision grammars including the English Resource Grammar, produces several forms of syntactic and semantic analyses of a sentence. However, these representations exhibit great structural complexity, i.e. utterances in ERG are modeled as large recursively nested feature structures and custom-designed underspecified logical forms.

Bilexical dependencies, i.e. binary head-argument relations holding between lexical units, is an attractive format since they on the one hand can reflect one of the most central decisions made explicitly or implicitly in most linguistic theories, e.g. the choice of a head, and on the other hand are formally and structurally relatively simple.

In this chapter we introduce a fully automated procedure for reduction of syntactic and semantic levels of native HPSG analyses into bilexical dependencies. Further we present a detailed qualitative and quantitative analysis of differences and commonalities of a range of different dependency formats.¹

3.1 Why bilexical dependencies?

Dependency representations have proven useful in diverse tasks of natural language processing and in a range of different downstream applications because bilexical dependencies can directly express predicate-argument relations, e.g. *Who did What to Whom Where and When?* The mainstream approach is incorporating bilexical dependencies as features for training statistical classifiers.

Some early adaptation of bilexical dependencies into system architectures include work of Gildea and Hockenmaier (2003) on semantic role labeling, Ding and Palmer (2005) on machine translation, Snow et al. (2006) on ontology learning, Wang et al. (2007) on question answering, Poon and Domingos (2009) on semantic search, Matsumoto et al. (2005) on opinion mining. More recent applications that exploit dependency relations are, among others, negation detection and resolution (Lapponi et al., 2012a; Mehrabi et al., 2015), speculation detection (Velldal et al., 2012), semantic role labeling (Foland and Martin, 2015), disfluency detection (Wu et al., 2015), sentiment recognition (Townsend et al., 2015) and opinion mining (Vilares et al., 2015).

¹Parts of this chapter are published in Ivanova et al. (2012) and Oepen et al. (2014).

3.2 Motivation

Bilexical dependencies find application in many tasks of computational linguistics as we showed in the beginning of the chapter. This practical utility motivates us to automatically derive linguistically grounded representations from the manually crafted grammar ERG. For some applications syntactic features derived from dependency trees are especially effective, while for others semantic information may be more favorable. ERG covers both syntactic and semantic levels therefore we derive two representations: syntactic and semantic dependency schemes, both grounded in the linguistic theory of HPSG.

We argue that it is worth preserving the particular properties of ERG in the dependency scheme rather than reducing the grammar structures to some existing dependency format (such as Stanford or CoNLL). One of the reasons to experiment with “native” dependencies is that downstream applications exploiting them differ in goals and integration method, which we demonstrated with the earlier examples. Variation in format gives a choice for the application developer to try several schemes and opt for the most appropriate one. Elming et al. (2013) showed that the choice of the conversion procedure has significant impact on the system performance in such NLP tasks as negation resolution, semantic role labeling, statistical machine translation, sentence compression and perspective classification. Even if the grammar structures are reduced to already known formats, the conversion itself will affect the performance of the application. Furthermore, in some cases there is no one-to-one correspondence between the structures of the grammar and the relations of the known dependency scheme, therefore the conversion would have to rely on heuristics which in turn introduce noise and result in error propagation. The creation of dependency resources involving as little conversion as possible was also a desideratum underlying the CoNLL 2007 shared task (Nivre et al., 2007a).

Dependencies are commonly used for cross-framework parser evaluation and comparison. To the best of our knowledge, prior to this work, there was no direct comparison of the HPSG PET parser with statistical parsers and the conversion to dependencies developed in the current chapter enables our work on cross-platform parser comparison in Chapter 5.

The Deep Linguistic Processing with HPSG Initiative (DELPH-IN) has produced a number of open-source manually and automatically annotated linguistic resources in the HPSG framework. Some of them were introduced in Chapter 2: DeepBank, Redwoods, WeScience. The resources present a range of domains and styles: Wikipedia texts, transcribed speech, topics about hiking and others. An automated parameterizable conversion to bilexical dependencies facilitates the availability of these resources to a broader natural language processing community.

To sum-up, the conversion of the ERG structures to binary word-to-word relations is motivated by the widespread integration of dependencies in various tasks of computational linguistics, the existence of dependency-based framework-independent parser evaluation methods and availability of the large linguistic resources annotated with gold-standard ERG that could be exploited by a larger community of researchers. We derive a new dependency format from ERG in order to preserve interesting properties of the grammar and avoid heuristics; moreover, there is no single standard in the field and variation might be beneficial. Introducing a new dependency format, we position it in the field by a structural comparison to several existing schemes both qualitatively and quantitatively. Such analysis makes explicit the information contained

in the treebank representations and shows a large variation across formats. In the following sections of the chapter we discuss related work, describe the conversion procedure and present a contrastive study of the various dependency representations.

3.3 Related work

In this section we give an overview of previous work dealing with the conversion from several other linguistic frameworks into dependency representations.

In the family of DELPH-IN HPSG grammars, experiments with conversion to dependencies were previously performed for the *ERG* and *Spanish Resource Grammar (SRG)* (Marimon, 2010). Zhang and Wang (2009) extracted bilexical dependencies from the derivation tree of *ERG* and in a similar spirit there was also developed a conversion procedure of the derivation tree of *SRG*².

For *Enju HPSG for English* (Miyao et al., 2004) dependency relations in the form of predicate-argument structures (PAS) are the native output of the *Enju* parser (Miyao and Tsujii, 2005). Miyao et al. (2008) proposed a pipeline to convert *Enju* dependency analyses into three other dependency formats. The parser output is first converted to Penn Treebank-style trees by tree structure matching and then further transformed by reduction procedures into Stanford Standard with the Stanford parser, into CoNLL with the Johansson and Nugues (2007) converter and into Head Dependencies with the Bikel (2004) software implementing the head detection algorithm of Collins (1997). This chain of reductions is imperfect and inherently introduces errors because in some cases PAS are inherently incompatible with PTB trees.

For *Alpino HPSG for Dutch* (Bouma et al., 2001) dependency relations are encoded in the grammar and offered as a standard output for the *Alpino* parser (Malouf and van Noord, 2004). There were several reasons for this design decision: firstly, dependency-based parser evaluation methods (Carroll et al., 1998) had already gained popularity, secondly, Collins (1999) parse selection models used dependency statistics, and thirdly, there was an ongoing project of dependency annotation of the Spoken Dutch Corpus (Oostdijk, 2000; van der Wouden et al., 2002). The dependency relations of *Alpino* are based on the guidelines of the Spoken Dutch Corpus and use co-indexing to express control relations. The dependency representation is integrated into the lexicon of the grammar as an additional layer of representation: a DT argument with the head and its dependents features are mapped to the elements of the subcategorization frame (see Figure 3.1).

Bilexical dependencies for *LFG* (Kaplan and Bresnan, 1982) can be obtained by reduction of one of the levels of representation in the grammar to dependency trees. *LFG* introduces two strata of syntactic analysis: c(onstituency)-structures and f(unctional)-structures. The former encode the word order and the hierarchical composition of words into phrases in a context-free phrase structure tree, while the latter provide grammatical functions like subject, object, etc. with morphological and semantic information in the shape of directed acyclic graphs. For each node in the c-structure tree there is a f-structure associated with it. Cetinoglu et al. (2010) describe the transformation of f-structures to bilexical dependencies. Applied modifications include: i) representation of each lexical item of a sentence in the f-structure since punctuation,

²http://www.iula.upf.edu/recurs01_mpars_uk.htm Accessed: 14 August 2015.

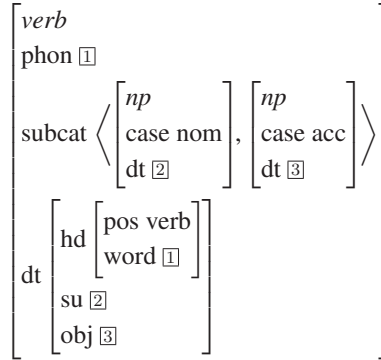


Figure 3.1: Simplified lexical entry from Alpino grammar for a finite transitive verb taking a direct object (Malouf and van Noord, 2004). The value of *dt* of the nominative *np* on *subcat* is identical to the value of the *su* dependent; the value of *dt* of the accusative *np* on *subcat* is identical to the value of the *obj* dependent

auxiliaries, particles and other tokens are not explicitly expressed in a form of a predicate; ii) elimination of the multi-headed constructions; iii) insertion of dummy dependencies for tokens without heads to avoid multiple roots or dropping the sentences for which this method is not applicable. In our conversion procedure for the ERG described below we encounter similar structural characteristics on the semantic layer of the grammar, however we avoid analogous changes assuming that graph parsing algorithms will be able to deal with DAGs. When development of such parsers matures enough, it will be possible to parse f-structures of LFG directly skipping the error-prone alterations of the f-structures.

For CCG (Steedman, 2000), a lexicalised grammar formalism based on combinatory logic, predicate-argument relations are explicitly defined in the lexicon and the root of the sentence is determined by the derivation (Bisk and Hockenmaier, 2013). The native output of the CCG parser (Clark et al., 2002) includes dependency tuples, e.g.

$$\langle \text{to}, \text{PP/NP}, 1, \text{report}, (\text{NP}\backslash\text{NP})/(\text{S}_{[\text{dcl}]}/\text{NP}) \rangle$$

that include the head word (to), its category (PP/NP), the argument slot (1), argument word (report), and the mediating category for long-range dependencies ((NP\NP)/(S_{[dcl]}/NP)) (Kim et al., 2012). Clark and Curran (2007a) proposed a transformation method of the native CCG dependencies to GR (see Section 2.4 for background about GR): the offline step is to map CCG predicate-argument relations to GR; at the processing time the parser output has to be modified to adjust the remaining differences between representations. During the mapping, several differences between the formats had to be accommodated. Firstly, the correspondence between the representations is many-to-many, therefore constraints were introduced to attain a one-to-one match. Secondly, all CCG relations are binary which is not always the case for GRs. For this reason there was added a special variable with corresponding constraints into the GR template that allows for the merging of several CCG dependencies into one GR relation}

during transformation. Thirdly, decisions about heads mismatched for some linguistic phenomena (e.g. complex verbs and relative pronouns) which required changes of head annotations in the CCG lexicon. Finally, some types of relations recognized by the CCG parser had to be ignored because they are absent from the GR-annotated corpus DepBank. The postprocessing rules concerned inter alia the treatment of coordination, ampersands, distinguishing arguments and adjuncts. Rimell and Clark (2009) developed a conversion of CCG relations to Stanford dependencies adopting the two-stage approach of Clark and Curran (2007a) of mapping CCG output to GR. For the first step of mapping between formats many of the solutions developed by Clark and Curran (2007a) were revised, and for the second step of postprocessing the new general rules were written to bring the transformed CCG dependencies closer to the Stanford scheme.

The review above suggests that there are several ways to derive bilexical dependency representation from a deep grammar. For the Enju grammar, dependency output is native and transformation to three other dependency representations is achieved via a chain of format conversions. For the Alpino grammar, a design decision was taken at a certain stage of grammar engineering to incorporate dependency relations natively in the lexicon in order to obtain dependency output for the Alpino parser. In the case of CCG dependencies are innate for the formalism and can be either extracted directly or transformed to more standard schemes via conversion. For LFG the native graph structures also express dependency relations but conversions are required to arrive at a more conventional dependency tree format.

In our work we derive bilexical dependencies from the ERG data representations. For conversion of ERG to syntactic dependencies we adopted an approach introduced by Zhang and Wang (2009) who performed dependency backbone extraction from the derivation tree of the *ERG*. The authors used the ERG version from July 2008 which differs from the more recent version 1212 employed in our experiments by having a smaller inventory of grammar rules and limitation on their arity (all the grammar rules were either unary or binary). A conversion to semantic dependencies is performed from Elementary Dependency Structures (EDS) (Oepen and Lønning, 2006), an MRS reduction to a variable-free dependency graph.

3.4 Conversion procedure

In this section we discuss the details of the dependency conversion from the syntactic and semantic levels of ERG into *DELPH-IN Syntactic Derivation Tree (DT)* and *DELPH-IN MRS-derived (DM)* dependencies respectively. Figure 3.3 exemplifies the syntactic layer of the ERG and Figure 3.4 illustrates the semantic representation that we reduce to dependencies for the Penn Treebank sentence:

- (23) *A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.*

The result of the conversions is presented in Figure 3.2 with syntactic dependencies above the sentence and semantic dependencies below the sentence.

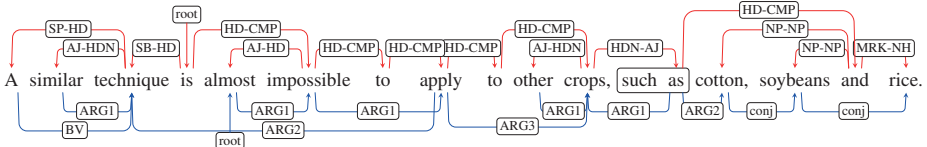
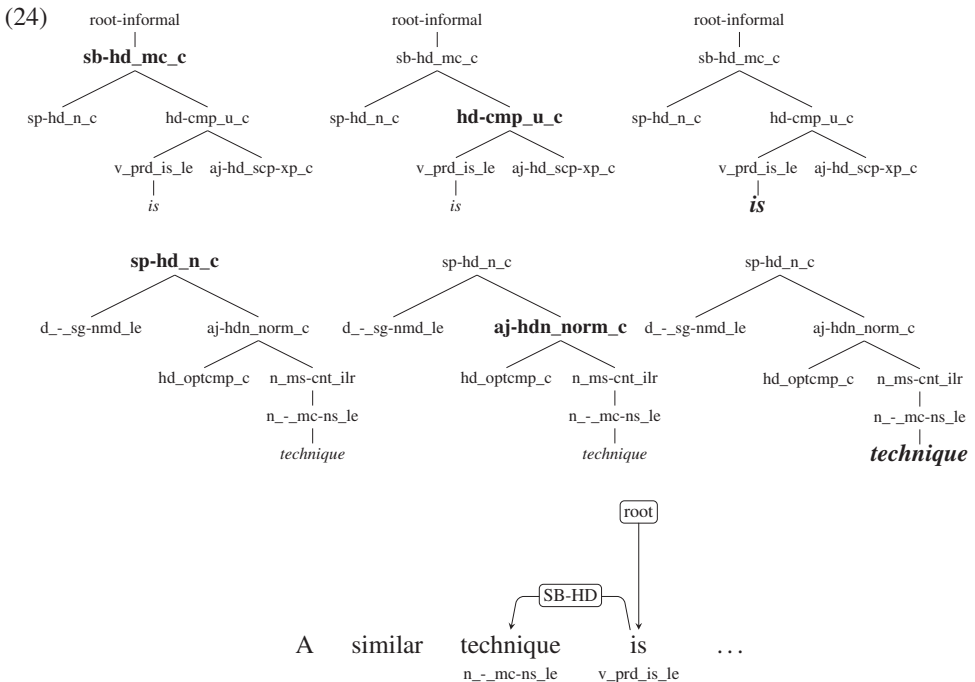


Figure 3.2: Dependency representations in DELPH-IN formats

3.4.1 Syntax: derivations to dependencies

As we recall from Chapter 2, the syntactic level of representation in ERG is embodied in a phrase-structure tree with HPSG constructions labeling internal nodes (e.g. `hd_optcmp_c`, ordinarily a phrase has to find all its complements before combining with a specifier,—a determiner, for nominal projections,—but this unary rule discharges an optional complement of the head), lexical types tagging preterminals (e.g. `d_-sg-nmd_le`, determiner singular not modified) and lexical tokens marking the leaves (see Figure 3.3). We transform this tree to the projective bilinear dependency representation called *DELPH-IN Syntactic Derivation Tree (DT)*.

During the conversion the tree is traversed and (a) the unary branches are eliminated, (b) the head daughter is looked up in the grammar configuration files for branching constructions. The “head” property is propagated down the tree to the lexical items. Example (24) explains step-by-step the process of identifying the root of the sentence and determining the first dependency.



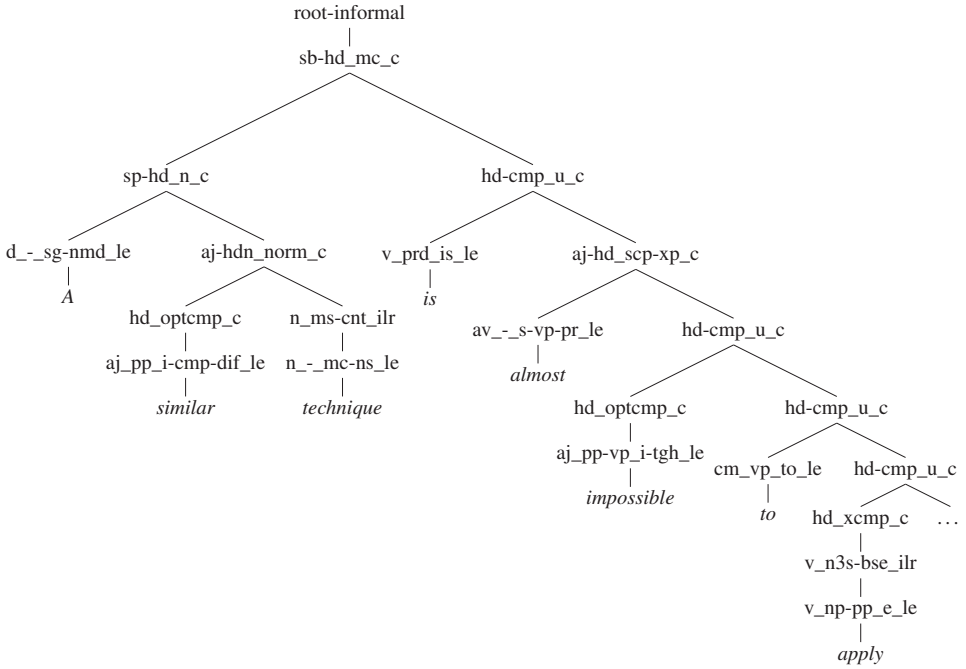


Figure 3.3: ERG syntactic derivation tree for the sentence “A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice.”

First, we skip the unary rule `root-informal` that suggests that the sentence is not perfectly edited (the sentence in Example 23 lacks the so-called Oxford comma in the tripartite coordination structure). For the branching subject-head construction in the main clause `sb-hd_mc_c` the conversion algorithm has to consult the head table in the configuration files. For this particular construction, it is also obvious from the first field of the rule name—`sb-hd`—that the daughter on the right (which is `hd-cmp_u_c` in our case) is the head, however not all the branching rules have an indication of the head position in their names. A fragment of the file with the head daughter specification for each rule is shown in Table 3.1 which describes how many child nodes each construction has (the grammar supports arbitrary arity of rules) and reports which daughter is the head counting left-to-right starting from 0. For `hd-cmp_u_c` (the head-complement construction in the unmarked clause) the left child node is the head, therefore, skipping the lexical type `v_prd_is_le` (a verb that takes predicative phrase as a complement) we derive that the lexical item “is” is the root of the sentence. The left daughter (`sp-hd_n_c`, head-specifier construction with the non-head,—the specifier,—as the semantic head) of the first branching rule (`sb-hd_mc_c`) is dependent on the root. The construction `sp-hd_n_c` has its head daughter on the left which is `aj-hdn_norm_c` (a nominal head construction with preceding normal, i.e. unmarked, adjunct). Finally, for `aj-hdn_norm_c` the child node on the right is the head, consequently skipping the unary `n_ms-cnt_ilr` (orthography-invariant, i.e. not changing orthography, inflectional rule for mass or count noun) and the lexical type `n_-mc-ns_le` (intransitive, i.e.

Rule	# of daughters	head daughter
sb-hd_mc_c	2	1
hd-cmp_u_c	2	0
sp-hd_n_c	2	1
aj-hdn_norm_c	2	1
...		

Table 3.1: ERG configuration file that describes how many child nodes each grammar construction has and which daughter is the head

not requiring complements, mass or count, no grammatically interesting, i.e. neither temporal nor animate, sort noun; this is a fairly typical type of noun), the procedure extracts the lexical item “technique”.

The label of the dependency link from “is” to “technique” is the generalization `sb-hd` of the first branching rule `sb-hd_mc_c`, and the lexical types of the word tokens become the part-of-speech tags. However it is our choice, which we will justify in Chapter 4, to generalize the ERG construction names and to preserve the whole name of the lexical type: the conversion procedure can produce different labels of differing granularity. The rest of the tree is built in a similar fashion (see Figure 3.2).

3.4.2 Semantics: logical form to dependencies

The semantic layer of ERG is expressed in the MRS formalism (Copestake et al., 2005) which was introduced in Chapter 2. However we perform the conversion to bilexical dependencies from Elementary Dependency Structures (EDS) (Oepen and Lønning, 2006): an MRS reduction that discards scope-related information. EDS already brings us close to a dependency format, however it does not necessarily represent a tree because it has the following properties:

- some lexical tokens are not expressed in the nodes of the graph (for example, semantically vacuous “to”);
- a node may have more than one incoming arc;
- several EDS nodes may correspond to a single token (e.g. for personal names);
- some nodes may not directly correspond to any surface token(s).

The complete EDS for the sentence in Example (23) is shown in Figure 3.4. Each graph node is shown on a separate line and each has an identifier (e.g. `_1`, `e9`) separated by a colon. Graph nodes are further labeled with semantic predicates (e.g. `_a_q`, `_similar_a_to`). Outgoing dependency arcs are listed in brackets, and each is labeled with a semantic argument role (e.g. `ARG1`, `ARG2`, `ARG3`) followed by an argument. For example, the third line depicts the node identified as `e9` and labeled with a predicate `_similar_a_to`. This node has one outgoing arc labeled `ARG1` pointing to the node `x6`.

```

{ e12:
  _1: _a_q(BV x6)
  e9: similar_a_to(ARG1 x6)
  x6: technique_n_1
  e12: almost_a_1(ARG1 e3)
  e3: impossible_a_for(ARG1 e18)
  e18: apply_v_to(ARG2 x6, ARG3 x19)
  _2: udef_q(BV x19)
  e25: other_a_1(ARG1 x19)
  x19: crop_n_1
  e26: such+as_p(ARG1 x19, ARG2 x27)
  _3: udef_q(BV x27)
  _4: udef_q(BV x33)
  x33: cotton_n_1
  _5: udef_q(BV i38)
  x27: implicit_conj(L-INDEX x33, R-INDEX i38)
  _6: udef_q(BV x43)
  x43: soybeans/nns_u_unknown
  i38: and_c(L-INDEX x43, R-INDEX x47)
  _7: udef_q(BV x47)
  x47: rice_n_1
}

```

Figure 3.4: ERG Elementary Dependency Structure

In the following we describe the conversion from EDS to the bilocal dependency scheme called *DELPH-IN MRS-derived dependency format (DM)* and explain the details of how the conversion procedure treats the EDS reflexes of various grammatical constructions. We introduce four types of ERG structures: *lexical*, *redundant*, *transparent* and *relational*. A lexical construction encompasses a predicate that corresponds to a surface token of a sentence; a redundant construction establishes several identical dependency relations; a transparent construction links two conceptually equal EDS nodes; a relational construction consists of a predicate that does not correspond to a lexical item but establishes dependency relations between its arguments. A predicate can be associated with multiple types of ERG structures, and in such a case the processing is sequential in the following order: redundant, transparent, relational, lexical. We will use our running example and sentences from the English MRS Test Suite³ to illustrate these four cases.

Lexical type Lexical predicates correspond to the surface tokens of the sentence. The basic conversion mechanism is straightforward: a unique identifier of a semantic predicate is considered the head, its arguments become dependents and argument roles are used as arc labels. This strategy applies to “regular” lexical relations, e.g. the ones that correspond to the actual word tokens. For example, `_other_a_1(ARG1 x19)` in Figure 3.4 contributes an ARG1 dependency between *other* and *crops*, because identifier `x19` denotes the EDS node `_crop_n_1` (see Figure 3.7e dependencies below the text). The vast majority of the lexical relations have a name that starts with an underscore sign and the others are listed in the configuration file (see Figure 3.5a) which defines parameters for the conversion.

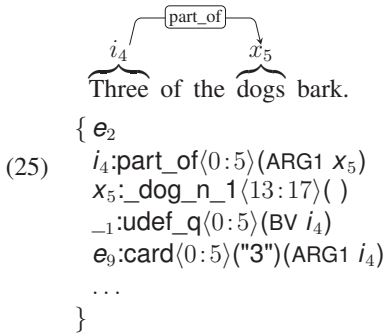
³<http://moin.delph-in.net/MatrixMrsTestSuiteEn> Accessed: 14 August 2015

<p>[lexical] card part_of person ... (a)</p>	<p>[redundant] ./ */ L-HNDL L-INDEX ./ */ R-HNDL R-INDEX (b)</p>
<p>[transparent] comp_equal ARG1 eventuality ARG1 nominalization ARG1 ... (c)</p>	<p>[relational] /_c\$/ L-HNDL R-HNDL /_c\$/ L-INDEX R-INDEX part_of ARG0 ARG1 ... (d)</p>

Figure 3.5: Excerpt from the ERG configuration file

Predicates that do not start with an underscore and are not included in the configuration file are ignored; furthermore, reflexive dependency relations (relations in which the head and the dependent correspond to the same lexical item) are purged from the dependency graph. Example (25) demonstrates that for the relation `part_of` a dependency link is established between the unique identifier i_4 of the predicate and the node x_5 corresponding to the words *three* and *dogs* respectively. The predicate `udef_q` is not analyzed because it does not start with an underscore and is not present in the configuration file. The predicate `card_q` should be analyzed as lexical according to the configuration file, but it introduces a reflexive dependency arc for the lexical item *three* which is excluded from the graph: the dependency relation is established between the nodes e_9 and i_4 that both denote the word *Three* because they have identical character span $\langle 0:5 \rangle$.

If the relation `part_of` did not belong to any other class of relations, the arc label would be ARG1, however predicates can belong to several classes, and `part_of` is also in the relational class, described below.



Redundant type Some EDS predicates introduce several equally directed dependency arcs with different labels between the two lexical items if the basic conversion algorithm used for lexical type is applied. The rules for the redundant relations in the configuration file permit us to

choose only one dependency arc for a word pair deterministically. MRS coordinate structures often encompass pairs of the right and/or left index and the handle that serve to differentiate between non-scopal and scopal arguments correspondingly (see Section 2.2.2 for the details about index and handle). This distinction is not conflated at the EDS level, however. In order to avoid redundancy of information, the index is purged whenever a complementary handle is present in the elementary predicate. Figure 3.5b shows how the rule is presented in the configuration file: the regular expression *./* does not impose any restrictions on the name of the relation and if L-HNDL is present, L-INDEX should be deleted, if R-HNDL is given, R-INDEX should be removed.

Example (26) illustrates the procedure discussed above: the EDS analysis of coordination *arrived and barked* includes left and right indexes and handles for designated nodes e_9 and e_{11} denoting lexical items *arrived* and *barked*, therefore the indexes can be ignored.

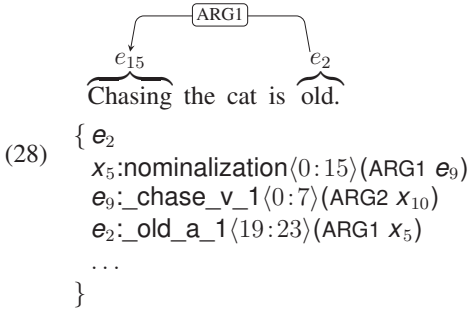
(26)
$$\begin{array}{l} \text{The dog } \overbrace{\text{arrived}}^{e_9} \text{ and } \overbrace{\text{barked}}^{e_{11}}. \\ \{ e_2 \\ e_9: _arrive_v_1 \langle 8:15 \rangle (\text{ARG1 } x_5) \\ e_2: _and_c \langle 16:19 \rangle (\text{L-INDEX } e_9, \text{R-INDEX } e_{11}, \text{L-HNDL } e_9, \text{R-HNDL } e_{11}) \\ e_{11}: _bark_v_1 \langle 20:27 \rangle (\text{ARG1 } x_5) \\ \dots \\ \} \end{array}$$

Transparent type The class of transparent relations includes semantic relations where in the underlying logic a referential instance variable is explicitly derived from an event, e.g. different types of nominalization. In terms of bilocal dependencies we want to conceptually equate the two EDS nodes involved. Identifiers of the predicates of this type are equated with one of their arguments. Figure 3.5c shows that the relations *comp_equal*, *eventuality* and *nominalization* are unified with their ARG1. In Example (27) the node x_{10} is equated to the node e_{15} corresponding to the lexical item “chasing”, which allows the converter to use the node e_{15} in place of x_{10} in the relation *_bother_v_1*. Note that if we assigned nominalization to the lexical class of predicates instead of transparent, the conversion would have led us to the reflexive dependency arc in this sentence because the predicates *nominalization* and *_chase_v_1* have the identical character span.

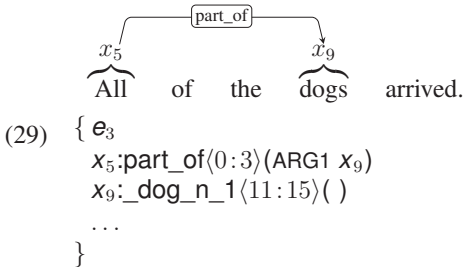
(27)
$$\begin{array}{l} \text{Browne 's } \overbrace{\text{chasing}}^{e_{15}} \text{ of cats } \overbrace{\text{bothered}}^{e_2} \text{ Abrams.} \\ \{ e_2 \\ e_{15}: _chase_v_1 \langle 9:16 \rangle (\text{ARG2 } x_{16}) \\ x_{10}: _nominalization \langle 9:16 \rangle (\text{ARG1 } e_{15}) \\ e_2: _bother_v_1 \langle 25:33 \rangle (\text{ARG1 } x_{10}, \text{ARG2 } x_{23}) \\ \dots \\ \} \end{array}$$

In Example (28) the node x_5 is equated with the node e_9 and as a result we obtain a dependency link between *old* and *chasing*. Note that in this case the *nominalization* predicate corresponds to a phrase *Chasing the cat is old* (in other words, what was nominalized in the grammar is the

complete verb phrase), therefore handling it as a lexical relation would have not led us to the word-to-word bilexical dependencies.

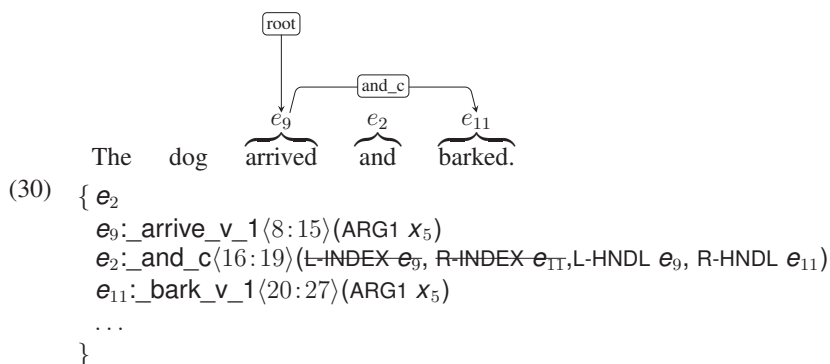


Relational type A bilexical dependency representation by definition does not contain nodes that do not correspond to a lexical item in the sentence. Some nodes of the EDS graph that do not correspond to surface tokens introduce relations that can be modified into bilexical dependencies by assigning the head role to one of the arguments, the dependent role to the other argument and using the predicate name as dependency label. The excerpt from our configuration file concerning these types of relations is shown in Figure 3.5d. For the predicate `part_of` its name is used as the dependency label, its `ARG0` becomes the head and its `ARG1` is considered as the dependent. In Example (29) `ARG0` corresponds to the node x_5 which in turn identifies the lexical item *All* and `ARG1` refers to the node x_9 representing the lexical item *dogs*.



All predicates of the relational type are at the same time associated with the transparent class of relations by default. The regular expression `/_c$/` from Figure 3.5d denotes any predicate with the name that ends with `_c`, e.g. any coordination structure. We opt for the analysis of Mel'čuk for coordination by making interchangeable conjunction nodes with their left arguments. The advantage of this analysis is that the incoming dependencies to coordinate structures receive the same category (e.g. verb) as they would have received without coordination. Furthermore, Schwartz et al. (2012) has shown that analyses that select the first conjunct as the head are easier to parse with statistical parsers than analyses that select the coordinating conjunction as the head of the coordination structure. One of the limitations of this approach is that it does not disambiguate a shared modifier and a private modifier of the first conjunct.

Example (26) illustrating redundant relations will be further processed as shown in Example (30) by equating L-HNDL with the conjunction, e.g. the node e_9 is further used in place of e_2 and the resulting dependency arc will be assigned from the word *arrived* to *barked*. Since the node labelled e_2 is the root of the EDS graph, the lexical item *arrived* corresponding to the node e_9 will become the root in the dependency representation.



3.4.3 Tokenization styles

For each sentence the conversion procedure can produce ERG- and PTB-style tokenization for the raw input, e.g. compare Figures 3.7e and 3.6 for our running example.

ERG tokenization conventions differ from the PTB in four aspects: (a) hyphens (and slashes) introduce token boundaries; (b) whitespaces in multiword lexical items do not always introduce token boundaries; (c) punctuation marks are attached as “pseudo-affixes” to adjacent words; (d) contracted negations are not broken up from the verb forms. Adolphs et al. (2008) provide some linguistic motivation for the approach chosen in ERG, for example, arguing that the PTB analysis of contracted negation is misleading from a linguistic point of view, since breaking up these constructions (e.g. “won’t” into $\langle wo, n't \rangle$) imposes pseudo-lexemes and leads to deriving nonexistent inflectional forms of verbs. Ma et al. (2014) provide some criticism of the PTB-inspired punctuation analysis arguing that the lexical heads of punctuation marks are vaguely defined and inconsistently annotated in the existing treebanks, and moreover that downstream applications, such as machine translation, usually utilize predicate-argument relations between words, rather than between words and punctuation symbols.

During the process of parsing with PET each token is assigned a character range consisting of the initial and final character positions in the source text, including spaces and punctuation. Such an approach can support several competing hypotheses about the number of tokens in a sentence (Flickinger, 2008). For example, the word “state-owned” is analyzed as a single token in PTB-style tokenization while in the ERG internal analysis it is split into two components. The character ranges are general enough to allow localization of the positions of the tokens in the text using both PTB- and ERG-style tokenization as shown in Example 31.

- (31) *Finmeccanica is an Italian state-owned holding company with interests in the mechanical engineering industry.*
PTB-style tokenization: <27:38> *state-owned*
ERG-style tokenization: <27:38> *state-*
 <27:38> *owned*

The character range <27:38> in Example 31 is used in PTB- and ERG-style tokenization to refer to the position of the item “state-owned” in the original sentence.

The PET parser builds an ERG analysis on top of the raw text. During the parsing process tokenization undergoes several modifications illustrated by Dridan (2013b) with the example shown in Table 3.2. Initially the raw text is treated with the Regular Expression-Based

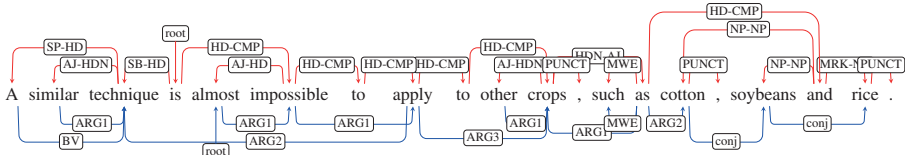


Figure 3.6: A WSJ sentence annotation using PTB-tokenization style

Raw	'Sun-filled', well-kept Mountain View.
	REPP
initial tokens	['], [Sun-filled], ['], [,], [well-kept], [Mountain], [View], [.]
	chart-mapping
internal tokens	['Sun-', [filled'], [well-], [kept], [Mountain], [View.]
	lexicon lookup
lexical tokens	['sun-', [filled'], [well- kept], [Mountain View.], [well-], [kept], [Mountain], [View.]

Table 3.2: Tokenization pipeline during parsing with PET (Dridan, 2013b)

Pre-Processing (REPP) toolkit integrated in the PET parser (see Section 2.2.2) to produce the tokenization that follows PTB convention. Subsequently, the chart-mapping mechanism is applied to the output of REPP described by token feature structures to adapt the tokenization to assumptions of the grammar. Thereupon prepared tokens are looked up in the lexicon and the candidate lexical items are added to the chart. As a result of parsing the output includes *initial tokens* that obey PTB-style tokenization, *internal tokens* that conform to ERG-style tokenization and ERG derivation tree leaves that correspond to the lexicon entries and comply with ERG-style tokenization.

Directly before the lexicon lookup the tokens are downcased and recorded in this form which causes a capitalization mismatch between the original raw text and the leaves of the derivation tree. The solution that allows us to restore the original case of the token in the vast majority of cases is to consult the **CASE** field of a derivation tree and if it is absent or does not provide enough information to refer to the internal tokenization by the character-based identifier. We will apply this technique for the data preparation in Chapter 5.

3.4.4 Output format

As an output format we have chosen the generic data format from the CoNLL 2008 shared task which is commonly used in the NLP community. It allows us to represent both syntactic and semantic dependencies for each sentence. The sentences in a file are separated by a blank line; each token is represented on a separate line and there are at least 11 fields separated by tabulation describing each token (see Table 3.3); comments are introduced with # symbol.

Straňák and Štěpánek (2010) note some weak points of the standard:

- there is no convention about how to add meta-information;

Field #	Field Description
1	Token counter, starting at 1 for each new sentence.
2	Word form or punctuation symbol (e.g. <i>company</i>)
3	Lexical entry from the derivation tree (e.g. <i>company_n1</i>) that represents lemma
4	Lexical rule from the derivation tree (e.g. <i>n_pp_mc-of_le</i>) that represents gold PoS
5	For ERG-style tokenization the same lexical rule as in field 4 (e.g. <i>n_pp_mc-of_le</i>), while for PTB-style tokenization a PTB PoS tag (e.g. NN)
6	–
7	Head index
8	Dependency relation
9	–
10	–
11	Semantic predicate derived from EDS
12- ...	Columns with semantic argument roles for each semantic predicate following textual order

Table 3.3: Fields in the output format for the ERG-derived syntactic and semantic bilexical dependencies

- the number of columns for a sentence corresponds to the number of predicates in the sentence and as a result the number of columns is high but the table is very sparse;
- representation of multi-layer annotations is not straightforward, especially when the surface tokens differ across the layers.

Despite the limitations of the format, it is widely used in the community and allows us to compactly represent information about syntactic and semantic dependencies in one file. We therefore choose to represent DT and DM in this form.

3.5 Contrasting analysis

In this section we will contrast the newly obtained dependency representations with several existing ones. We pursue the goal to shed some light on the differences and commonalities of various schemes and explore how the ERG-derived formats relate to other representations. Figure 3.7 displays the syntactico-semantic dependencies in the nine formats (syntactic on top and semantic below) introduced in Section 3.5.1 which we will compare qualitatively in Section 3.5.2 and quantitatively in Section 3.5.3.

3.5.1 Variation in dependency representations

Dependencies are binary asymmetrical relations expressing direct predicate-argument relations between lexical units. Practical benefits of dependency syntax are found in its rather intuitive structure showing links between words, and existence of linear-time algorithms for

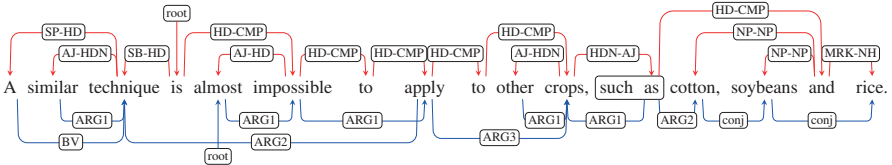
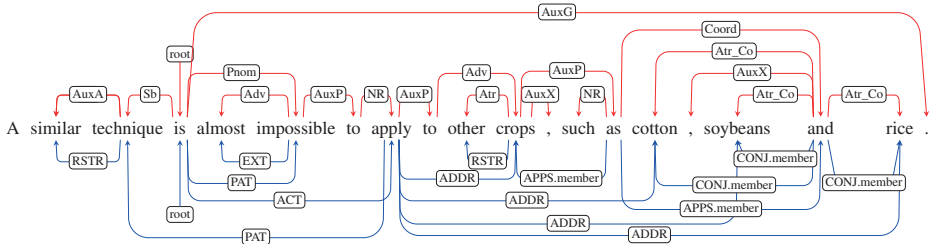
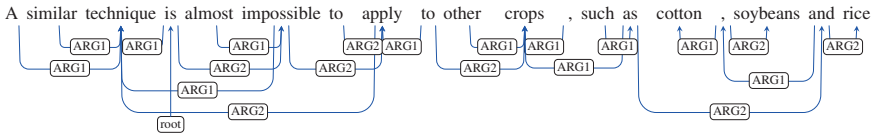
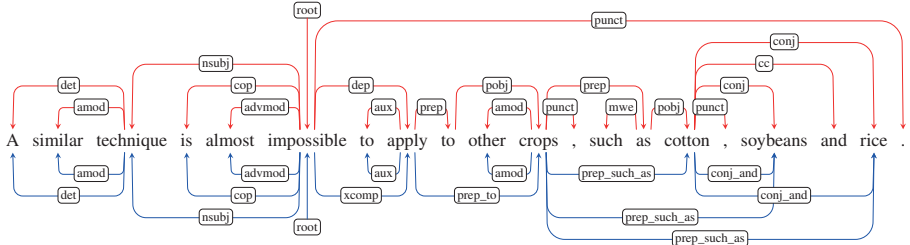
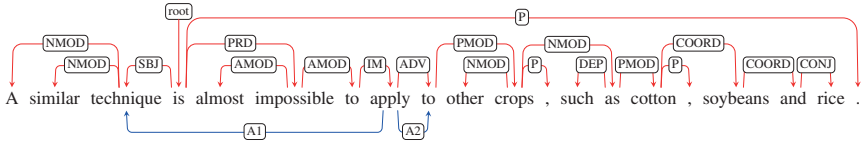


Figure 3.7: Dependency representations in (a) CoNLL, (b) Stanford (c) Enju, (d) Prague, and (e) DELPH-IN formats

analysis. A variety of different dependency formats has emerged in the past years and in this section we will present those that we have chosen for our study and start contrasting them by

some of their graph properties. As we recall from Chapter 2, it is commonly assumed that the dependency structure of a sentence is a tree, e.g. a directed graph, conforming to the following conditions (Nivre et al., 2007b):

- every token in the sentence constitutes a node in the graph;
- there is one designated root node, conventionally numbered as 0;
- the graph is (weakly) connected;
- every node in the graph has at most one head;
- the graph is acyclic.

Dependency types are usually expressed in the graph as arc labels.

A broad range of schemes usually accommodate these constraints but differ in choices of head for various phenomena and sets of dependency labels, while more linguistically-motivated representations often do not adhere to the above-mentioned requirements. We may distinguish between formats that take a largely *functional* view on head status—e.g. functional elements like auxiliaries, subjunctions, and infinitival markers are heads—and more *lexical* or content-centered approaches where the lexical verbs or arguments of the copula are heads. The dichotomy between lexical and functional categories is theoretically motivated and has played an important role in generative theories of syntax such as Principles and Parameters, Government and Binding, Minimalism, though the distinction is not always rigid and the borderline cases are described as semi-lexical categories (Corver and van Riemsdijk, 2001). Table 3.4 provides a summary of the head type for the formats we are focusing on in this chapter and Figure 3.7 shows a sentence annotated with these nine formats and illustrates drastic differences between them⁴.

In the following, we will discuss several formats which appear in the literature reasonably frequently and have been exploited in parsing and other tasks: syntactic and semantic CoNLL dependencies, basic and standard versions of Stanford dependencies, Enju predicate-argument structures and analytical and tectogrammatical layers of Prague linguistic annotations. One of the motivations for our choice is also the availability of the annotations in all of these representations of the same large linguistic resource, e.g. the Wall Street Journal (WSJ) portion of the Penn Treebank.

CoNLL Syntactic Dependencies (CD) This dependency representation was already introduced in Section 2.5. The scheme was utilized in several CoNLL shared tasks dedicated to parsing into word-to-word dependencies and was obtained via conversion of PTB with the LTH constituent-to-dependency conversion tool for Penn-style treebanks (Johansson and Nugues, 2007). This conversion relies on head finding rules (Collins, 1999) and the grammatical function labels, empty categories and named entities present in the PTB annotation. The format supports long-distance phenomena such as *wh*-movement and topicalization often expressing them with non-projective dependency links. The tool also takes advantage of the annotation

⁴The annotation of coordination in Enju PAS in Figure 3.7 differs from the analysis shown in Ivanova et al. (2012) because the gold standard annotation for this sentence was corrected by Prof. Yusuke Miyao.

of the internal structure of base noun phrases in PTB added by Vadas and Curran (2007) (see Figure 2.13 in Section 2.2.2). The dependency structures in this format obey the general conditions mentioned at the beginning of this section and the syntactic heads are largely functional, e.g. infinitival markers and auxiliary words are chosen as heads.

CoNLL PropBank Semantics (CP) This dependency representation was created in conjunction with the CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies (Surdeanu et al., 2008). The format was produced by conversion of PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) annotations that provide argument structure for verbs and common nouns in PTB, into a dependency analysis. The conversion does not exploit PTB syntactic constituents but relies on CoNLL Syntactic Dependencies for compatibility. Dependency structures are generated by a heuristic process that assigns the head of a semantic argument to the token inside the argument boundaries whose head is a token outside the argument boundaries (Surdeanu et al., 2008). The heuristics work accurately for a vast majority of arguments with some exceptions that receive further treatment. For example, a semantic argument is split into a sequence of discontinuous arguments if the heuristics mistakenly assign several syntactic heads to it. The representation does not adhere to the formal constraints posed above: it lacks a dummy root node, the graph is not connected, and the graph is not acyclic. The choices with respect to head status are largely lexical.

Stanford Basic (SB) The Stanford schemes (de Marneffe et al., 2006), briefly presented in Section 2.5, are designed to extract dependency relations from phrase structure parses. Originally the representation was provided as an additional output format for the Stanford parser (Klein and Manning, 2003). The grammatical relations are organized into a hierarchy with the more general relations on top and more specific ones further down. The conversion is a two-step process: dependency extraction and dependency typing. Initially a phrase structure analysis is built for a sentence with some parser trained on Penn Treebank and then the head-finding rules similar to Collins (1999) rules are applied to identify the head for each constituent. Lexical heads are favored in this format, therefore most of the relations are established between content words. The second step involves pattern-based labeling of the extracted dependencies with grammatical relations. The search algorithm compares patterns against the parse tree and chooses the most specific grammatical relation from the detected matching patterns. Each word token corresponds to a node in the resulting dependency tree which is a single-rooted directed acyclic graph.

Stanford Collapsed Dependencies (SD) Stanford Dependencies also come in a so-called collapsed version where some pairs of dependencies are merged into a single node purging the word in between from the graph. The collapsed words which are primarily prepositions, conjunctions and possessive clitics are used in the label names as shown in Example (32).

- (32) *crops such as cotton , soybeans and rice ...*
 prep_such_as(crops, cotton)
 conj_and(cotton, soybeans)
 conj_and(cotton, rice)

Department 's index

poss(index, Department)

The collapsed representation does not meet the formal graph criteria mentioned above: it is not connected, since not all tokens in the sentence are represented in the graph, a node may have more than one head, and there may also be cycles in the graph.

Enju Predicate-Argument Structures (EP) The Enju system is a robust, statistical parser obtained by learning from a conversion of the PTB into HPSG (Miyao and Tsujii, 2005). Enju outputs *predicate-argument structures* (PAS) that describe semantic relations of words, phrases and clauses in a sentence. The representation of the dependency types in Enju is the same as that of PropBank (Kawahara et al., 2013) (ARG1, ARG2 etc.) and the graph structure does not obey the constraints mentioned above. Enju PAS primarily aims to capture semantic relations and it encodes most types of syntactic modifiers as predicates. EP chooses auxiliary verbs, copula and subjunctions as heads, i.e. it tends to have functional heads.

Prague Analytical Tree (PA) The Prague Czech-English Dependency Treebank (PCEDT) (Čmejrek et al., 2004) provides Prague Dependency Treebank-style annotations (Hajič, 1998) grounded in Functional Generative Description (Sgall et al., 1986) for the WSJ corpus. The analytical layer of the markup describes the surface syntax of a sentence. Dependency representations of the English part of PCEDT are produced automatically from PTB by a conversion procedure similar to the one discussed in Xia (2001). First, terminal nodes of the phrase structure tree are transformed to nodes of the dependency graph and then dependencies between the nodes are established recursively starting from the root using Jason Eisner's head assigning scripts (Eisner, 2001). The nodes for traces are removed from the graph and their child nodes are attached to the trace parent nodes. In the analytical level of representation, constructions that require structural transformation different from the algorithm described above are coordination and apposition. Each node of the analytical tree is assigned a surface syntactic attribute called *analytical function* that describes its relation to the head. The rule-based process of labeling nodes with analytical functions relies on the information about function and PoS tags of the original PTB tree and the local context of the dependency tree, e.g. the rule $m\text{POS}=\text{MD}|p\text{POS}=\text{VB}|m\text{AF}=\text{AuxV}$ labels a modal verb headed by a verb with the analytical function AuxV. The markup of coordinations, appositions and prepositional phrases is tackled separately. The analytical tree is consistent with the general dependency graph constraints that we described above.

Prague Tectogrammatical Tree (PT) The tectogrammatical layer of sentence representation in the theoretical framework of the Functional Generative Description is a level of linguistic (literal) meaning. The tectogrammatical annotation is manually built on top of the analytical layer trees. Unlike the analytical layer, functional words (prepositions, punctuation marks, determiners, subordinating conjunctions, certain particles, auxiliary and modal verbs) do not constitute graph nodes of the tectogrammatical trees but become attributes of their heads. In the tectogrammatical level, all valency frames must be realized, therefore the tree includes additional nodes restored using traces that do not correspond to the tokens of the sentence. Each node of

the tectogrammatical tree is assigned a *functor* and a set of *grammatemes*. Tectogrammatical functors are semantic-syntactic labels that describe dependency relations (e.g. Actor, Patient, Effect); grammatemes are semantically-oriented morphological categories such as tense, number, degree of comparison, and some of them describe derivation information (Razímová and Žabokrtský, 2005). The rules that assign functors use PTB PoS tags, function tags and lemma. The markup of morphological grammatemes exploits PTB PoS tags. The tectogrammatical tree does not agree with the constraints mentioned in the beginning of the subsection because some tokens are excluded from the graph and additional nodes without surface realization are added. PT bilexical dependencies shown in Figure 3.7d and used further in our analysis are extracted from tectogrammatical trees using a conversion procedure described in Miyao et al. (2014). These dependencies were employed in the Broad-Coverage Semantic Dependency Parsing (SDP) shared tasks (Oepen et al., 2014, 2015). PT graphs encode only a subset of the original tectogrammatical annotation, excluding among other things nodes representing elided elements, coreference links and grammatemes.

3.5.2 Qualitative analysis

As a basis for a qualitative comparison we use WSJ sentences annotated in the 9 target dependency schemes. We have chosen most of the CD, CP, SD, EP annotations from the collection dubbed PEST (Parser Evaluation shared task)⁵ developed for the shared task on comparing representations for grammatical analysis (Bos et al., 2008) at a workshop on Cross-Framework and Cross-Domain Parser Evaluation of the 2008 Conference on Computational Linguistics (COLING). In addition we used the Stanford parser to convert gold-standard annotations of the same WSJ sentences to SB format. Applying the conversion procedure described in the previous section to the DeepBank we obtained DT and DM annotations. Further we extracted PA markup from PCEDT and collected modified Prague tectogrammatical layer representation (PT) from the data employed in the Broad-Coverage Semantic Dependency Parsing (SDP) shared tasks. We have chosen SDP data instead of the original tectogrammatical annotations from PCEDT in order to have the set of graph nodes equivalent to the set of surface tokens across all the formats in question. As we have already mentioned above, SDP uses only a subset of the tectogrammatical annotation, e.g. elided elements, coreference and grammatemes are excluded while functional and punctuational tokens hidden in PCEDT tectogrammatical layer are on the contrary included in the graph. As a result we have 24 sentences from PEST and some additional WSJ sentences represented in 9 aligned formats,—CD, CP, SB, SD, EP, PT, PA, DT, DM.

Figure 3.7 visualizes a range of structural variations across representations. Using our running Example (23) we will examine several linguistic phenomena where the formats differ. One of the points of disagreement between the schemes is the nature of the head: some formats can be classified as more functional and others as more substantive as we have already mentioned in Section 3.5.1. Table 3.4 summarizes some of our observations from Section 3.5.1 about the head status, graph structure and connectivity of the formats. In the following we will examine

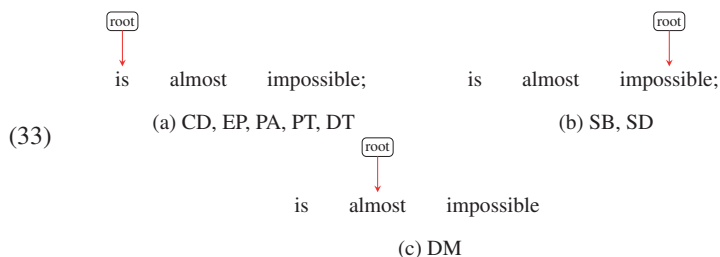
⁵The WSJ section of the PEST collection contains 25 sentences, however one sentence lacks annotations in the EP format. The full PEST collection includes another three dozen sentences from other corpora, apart from WSJ, but the annotations are not available for all the formats and in some cases they are not manually validated.

Description		Head	Tree	Connected
CD	CoNLL Syntactic Dependencies	F	+	+
CP	CoNLL PropBank Semantics	S	–	–
SB	Stanford Basic Dependencies	S	+	+
SD	Stanford Collapsed Dependencies	S	–	–
EP	Enju Predicate–Argument Structures	F	–	+
PA	Prague Analytical Tree	F	+	+
PT	Prague Tectogrammatical Tree	S	+	–
DT	DELPH-IN Syntactic Derivation Tree	F	+	+
DM	DELPH-IN Minimal Recursion Semantics	S	–	–

Table 3.4: Summary of dependency formats, where the columns labeled *Head* indicate the head status—functional (F) vs. substantive (S), *Tree* whether or not structures are acyclic trees, and *Connected* whether or not all tokens are weakly connected

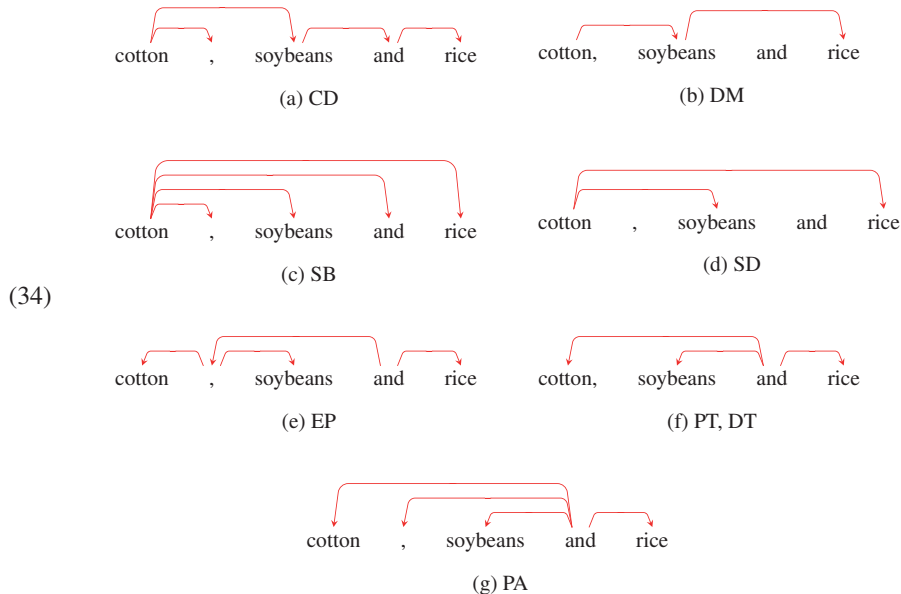
several linguistic structures and contrast the formats with respect to these. Some formats do not provide annotation for some linguistic constructions, in such cases we do not show them in the examples below.

Root/copula Example (33) displays the three possible choices for the root for our example sentence. Most of the schemes choose the auxiliary verb *is* (see Example (33a)), while the more substantive Stanford formats have the predicative adjective *impossible* as the root (see Example (33b)) and, finally, the more semantic DM scheme chooses the scopal operator *almost* as the root (see Example (33c)). The scopal operator defines a scope which encompasses the copula *is* and the predicative argument *impossible* and affects the truth condition of the sentence.



Coordination Dependency relations are asymmetrical in essence (if *A* is the head of *B*, then *A* does not depend on *B*) therefore the analysis of symmetric constructions like coordination is unavoidably challenging. In Example (34) we can see striking disagreement of the formats with respect to the analysis of conjunction. CP has no analysis for coordination structures, since it only analyzes main arguments in the sentence. The CD analysis shown in Example (34a) and the DM analysis depicted in Example (34b) exhibit Mel’čuk-style analysis with a chain of dependencies headed at the first conjunct, but the latter scheme excludes the coordinating conjunction from the dependency tree since this element is functional. In both Stanford schemes the first conjunct is the head and the other conjuncts depend on it as shown in Examples (34c,d),

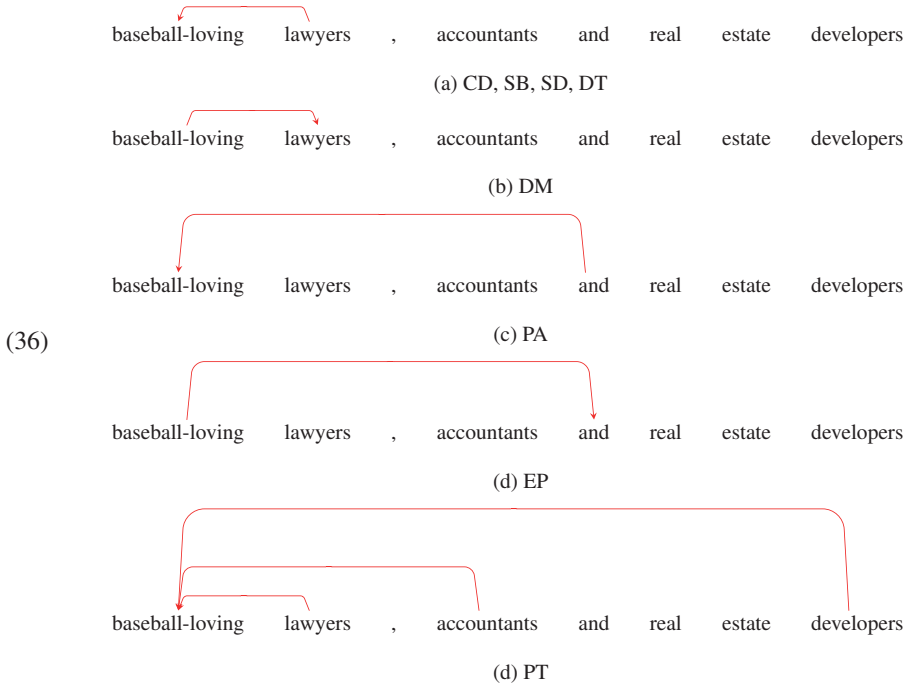
though the SD scheme excludes functional elements similarly to DM. In the PT, DT and PA the coordinating conjunction is the head for all conjuncts as shown in Examples (34f,g). In EP the last coordinating conjunction is the head of the whole structure and each coordinating conjunction (which can be represented with a punctuation mark) takes left and right conjuncts as its arguments as illustrated with Example (34e).



An interesting property of coordination is found in the attachment of a shared modifier. In our running sentence we do not have such a modifier, therefore we will illustrate this structure with another PTB sentence:

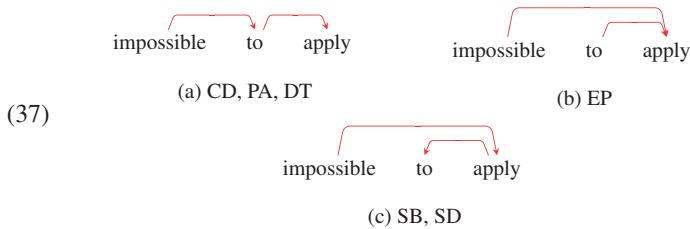
- (35) *It's one more for the baseball-loving lawyers, accountants and real estate developers who ponied up about \$1 million each for the chance to be an owner, to step into the shoes of a Gene Autry or have a beer with Rollie Fingers.*

The conjuncts of the coordination construction *lawyers, accountants and real estate developers* share a modifier *baseball-loving* in this example. Most schemes (CD, SB, SD and DT) attach the shared modifier to the first conjunct as shown in Example (36a) and one (DM) (see Example (36b)) attaches the first conjunct to the shared modifier. In such cases it is not disambiguated in Mel'cuk style whether the modifier is shared or depends only on the first conjunct. EP, PT and PA, on the other hand, show explicitly whether a modifier is shared or private as illustrated with Examples (36c,d): in PT all conjuncts are heads of the modifier since multiple heads are allowed in this format, in PA and EP there is a dependency between the shared modifier and the coordinating conjunction.



Popel et al. (2013) give a systematic survey about coordination topology in treebanks that adopt Prague Dependency Treebank-, Mel’čuk- and Stanford-style analyses for different languages. In addition to the differences that we observe for English, e.g. variation in attachment of shared modifiers, coordinating conjunction and punctuation and diversity of label variants, on a cross-language scale there is no general standard whether the leftmost or rightmost conjunction/punctuation/conjunct should be the head of the coordination structure, e.g. the Persian Dependency Treebank (Rasooli et al., 2011) uses the rightmost head for coordination of verbs and leftmost heads for other types of coordination.

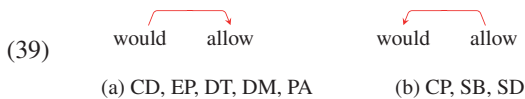
Infinitival constructions The infinitive *to apply* receives different interpretations in the formats as illustrated with Example (37). The infinitival marker depends on the main verb only in the Stanford formats, whereas CD, PA and DT regard it as a head. In EP infinitival markers and other auxiliaries that we analyze below (auxiliary verbs, adverbial portions of phrasal verbs, negators, prepositions) function as predicates, though they do not always serve as local top nodes (e.g. the semantic heads of the corresponding sub-graphs): the infinitival marker in Example (37b) takes the verb “apply” as its argument, but the predicate “impossible” links directly to the verb. CP, PT, DM exclude infinitival “to” from the graph since it does not bear semantic meaning on its own.



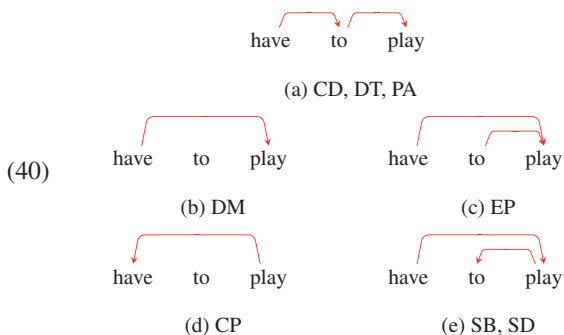
Complex verbs Auxilliary verbs in complex verb phrases are treated by the formats analogously to infinitival markers in infinitive constructions, with the only exception of PA that selects the infinitival marker as a head of the lexical verb but the auxiliary verb as dependent of the main verb (cf. Examples (38a) and (38b)).



It is interesting to note that for the verb groups that are composed of a verb and a modal verb the picture is different as the more semantic formats, except PT, do not ignore the modal. The general trend is shown in Example (39).

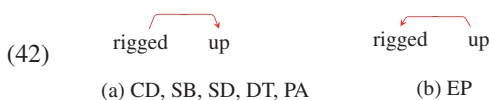


For a more complex case of the modal verb “have to”, there is more variation in annotation, as illustrated with Example (40).



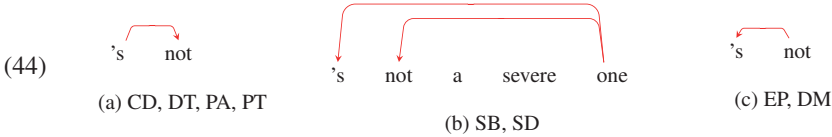
Verb-particle constructions Verb-particle constructions are analyzed as semantically vacuous in DM, PT and CP and in all other formats except EP (see Example (42b)) they depend on the main verb as shown in Example (42a).

(41) *After discovering that the hacker had taken over the dormant account of a legitimate user named Joe Sventek , he rigged **up** an alarm system , including a portable beeper , to alert him when Sventek came on the line.*



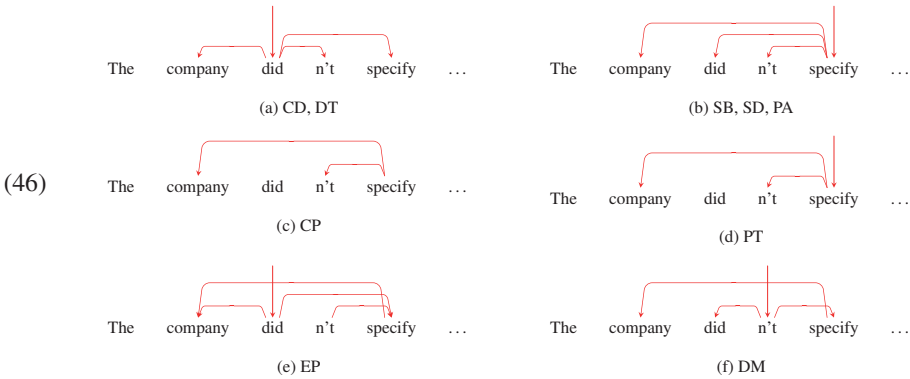
Negators In CD, DT, PA and PT negators are dependent on the verb (see Example (44a)) and in Stanford formats on the nominal predicate (see Example (44b)), while in EP and DM negators function as heads (see Example (44c)). From a syntactic point of view negation marker can be seen as a modifier of a predicate, while semantically negator can be selected as a governor since it affects the truth value of a statement.

(43) *It is a considerably delayed reaction and it's **not** a severe one at all, " she added*

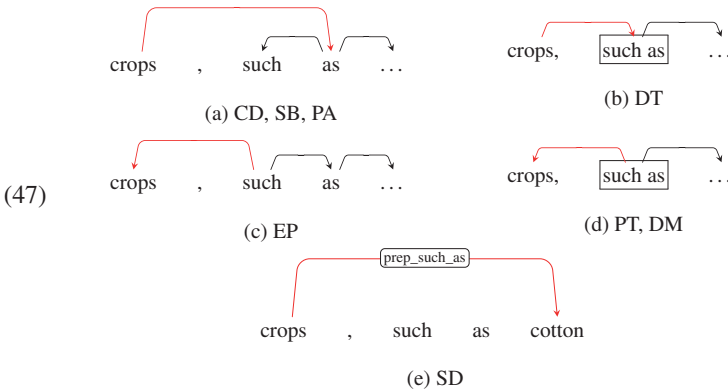


Example (46) shows complex dependency relations in a sentence with a negation in more detail and it also illustrates that in CP negators are arguments of the verb predicates.

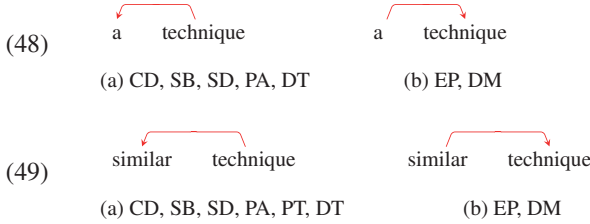
(45) *The company did **n't** specify reasons for the strong earnings gain .*



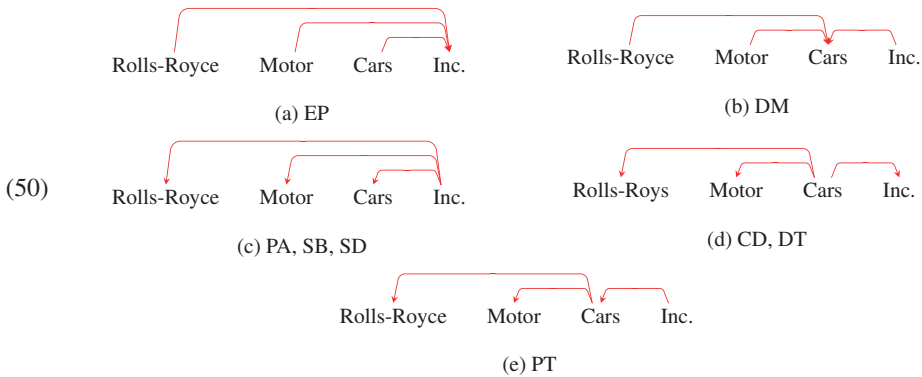
Prepositional phrases In CD, SB, PA and DT prepositions are dependents as illustrated by (*such*) *as* in Examples (47a,b); in EP, PT and DM, prepositional modifiers are heads (see Examples (47c,d)); and SD “collapses” prepositions to yield direct relations between the nominal head of the preposition (*crops*) and its internal argument (see Example (47e)).



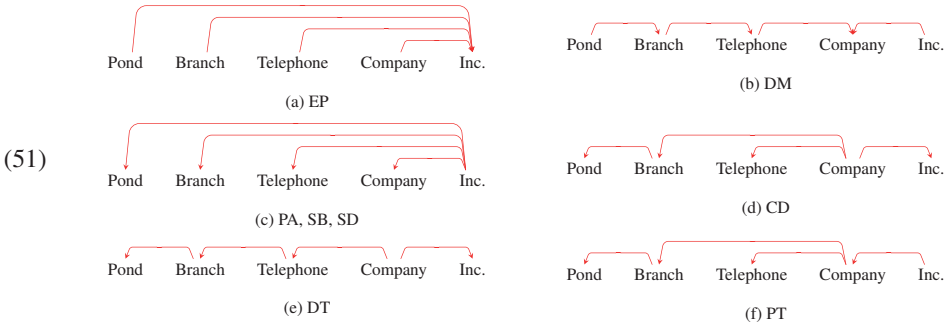
Noun phrases In noun phrases such as *a similar technique* one can treat the determiner and attributive adjective as dependents of the noun, which is what we find in the CD, SB, SD, PA and DT schemes (see Examples (48a) and (49a)). Alternatively, one may consider the noun to be a dependent of both the determiner and the adjective, as is the case in the schemes EP and DM deriving from predicate logic (see Examples (48b) and (49b)). In PT articles are not represented as nodes in the dependency graph, while for adjective modifiers the noun functions as the head.



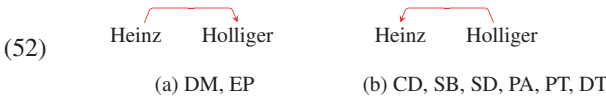
Complex nouns The analysis of complex nouns can differ greatly across formats because some annotations reflect the inner structure of the compounds while others provide a simplified analysis. As we discussed in Chapter 2, CD is derived from the PTB patched with the NP bracketing by David Vadas recovering NP structure. The HPSG formalism also provides non-trivial analysis of complex nouns, and the Enju Treebank from which EP is derived, anteceded the bracketing of Vadas and Curran (2007). Example (50) illustrates different versions of dependency analysis of a company name. Most of the formats, i.e. SB, SD, PA, PT, EP, DT, DM provide very generic dependency labels, i.e. “nn”, “nn”, “Atr”, “NE”, “noun_ARG1”, “NP-HDN” and “compound-name” correspondingly. In CD posthonorifics such as “Inc.” are annotated using label “POSTHON”.



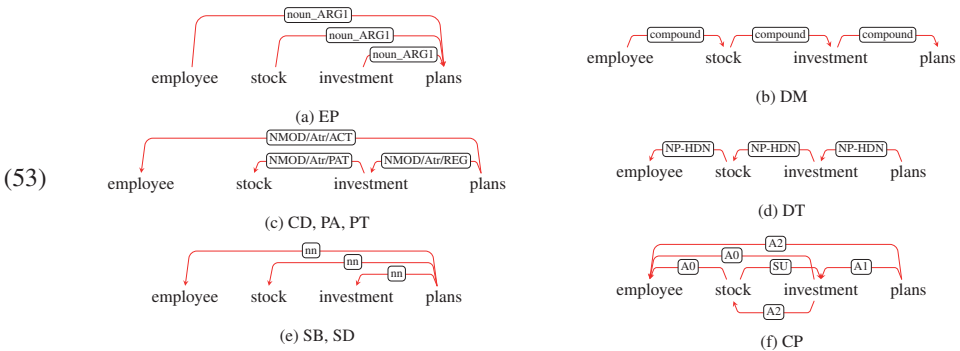
Interestingly, another company name can be analyzed differently within the same dependency framework, e.g. for CD, DT, PT and DM analyses in Example (51) differ from analyses in Example (50). One of the main differences for CD, PT, DT and DM is due to the fact that “Pond” modifies “Branch” and not “Company” in Example (51). For SB, SD, EP and PA analyses in Examples (50) and (51) are identical.



For less complex personal names consisting only of the first name and last name of a person, there is a clear division between the formats, see Example (52).



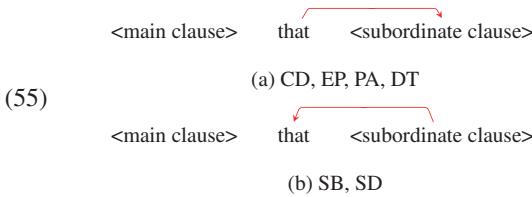
For the complex noun groups that do not represent named entities as shown in Example (53) we again observe diversities not only between different formats but also differences within one framework compared to analysis of named entities. For example, CP does not provide analyses of the complex named entities shown in Examples (50), (51) and (52) above, but it gives a detailed analysis of the non-named entity noun phrase in Example (53). Only CP and PT use specific labels for complex noun phrases while other formats opt for generic ones. In PT representation “employee” is an ACTOR of “plans”, “stock” is a PATIENT of “investment” and the dependency from “plans” to “investment” is labeled with REG (“with regard”), which makes the semantics of the phrase rather clear: employee is the initiator of the plans with regard to investment in stock. In CP “employee” is ARG0 of “stock” and “investment” and ARG2 of “plans”; “stock” is ARG2 of “investment”; “investment” is SU (segment unit) of “stock” and ARG1 of “plans”. ARG0 is usually used to classify agents, ARG1 - patients and themes, ARG2 - indirect objects (Meyers et al., 2004). This means that analysis of the phrase in CP, PA and PT expresses the semantics that there are *plans of investment in stock initiated by the employee*. From the DM analysis one can potentially interpret that the phrase is about *investment plans in a stock for employees* (in contrast to “common stock”) because of the dependency relation between “employee” and “stock”. However, such fine distinctions are very hard to make without context and expert knowledge.



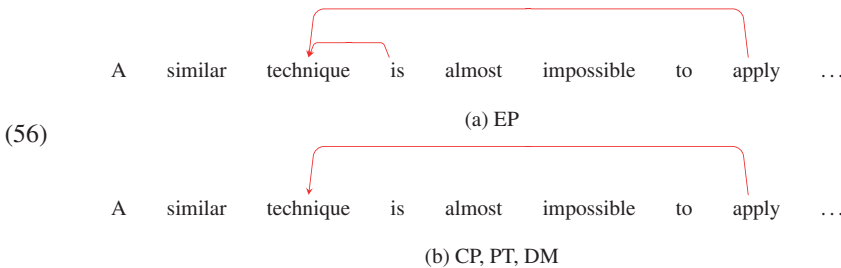
Subjunction We will illustrate the attachment of subjunctions on the PTB with Example (54) where the subjunction *that* connects the main (*Management and labor worry*) and subordinate (*the gap makes U.S. companies less competitive*) clauses in one sentence.

(54) *Management and labor worry that the gap makes U.S. companies less competitive*

We find that only the Stanford formats choose the word *that* governs the subordinate clause as the head of the subjunction (see Example (55b)) while the more functional CD, EP, PA and DT formats have the opposite direction of this dependency arc (see Example (55a)). In the CP, PT and DM graphs subjunctions do not correspond to a separate node.



Tough adjective Our running example invokes the so-called *tough* construction, where a restricted class of adjectives (*impossible* in our case) select for infinitival VPs containing an object gap and, thus, create a long-distance dependency (Rosenbaum, 1967; Nanni, 1980). In the dependency analyses in Figure 3.7 we observe three possible heads for the noun *technique*, viz. *is* (CD, EP, PA and DT), *impossible* (SB and SD), and *apply* (CP, EP, PT and DM). The long-distance dependency between *technique* and *apply* is marked only in the more semantic schemes: EP, CP, PT and DM, see Examples (56a,b).



3.5.3 Quantitative analysis

Quantitative analysis helps to further contrast the dependency schemes and explicate whether interconversion between some subsets of formats may be possible in general. An objective similarity metric shows the degree of overlap between the different representations.

For the quantitative analysis we use the *Jaccard similarity* measure, which is a statistic for comparison of two sample sets, and the *unlabeled F1 score*, commonly applied in parsing to measure the similarity of the parsing results to the gold standard. Our choice is motivated by the properties of these metrics: symmetry (e.g. CD should be as similar to SB, as SB to CD) and appropriateness for binary classification (pairwise comparison). Jaccard similarity is a monotonically increasing function of F1 (Lipton et al., 2014) therefore we expect them to show similar trends. We use two metrics to verify the results of one another.

We computed the similarities on the 29,867 sentences from the WSJ corpus, annotated in 9 different formats. The data represented in CD format is obtained by the conversion of the gold PTB trees using the software of Johansson and Nugues (2007); the CP representation is taken from the gold-standard data used in CoNLL 2008 shared task; EP comes from the data collection of SemEval-2014 shared task on Semantic Dependency Parsing (SDP); PA is produced by the automatic conversion of PCEDT with the Treex package; PT comes from the SDP corpus; DT and DM are derived by an automatic conversion from the gold HPSG analyses of the DeepBank (DM is also included in SDP). All 9 formats are aligned in PTB tokenization, and for this purpose we collapsed multiword expressions in one token in CP representation, filtered out sentences that miss punctuation in the middle of the sentence in the EP format which are typically complex sentences with citations as shown in Example (57), and filtered out sentences for which PTB tokenization in the gold DeepBank analyses differs from the tokenization in the PTB corpus which is commonly caused by apostrophes, lists, three dots and acronyms as shown in Example (58).

(57) PTB tokenization:

“ What sector is stepping forward to pick up the slack ? ” He asked .

EP tokenization:

“ What sector is stepping forward to pick up the slack ” He asked

	PTB tokenization	DeepBank tokenization
	The '82 Salon is \$ 115 .	The ' 82 Salon is \$ 115 .
(58)	1 . Buy a new Chevrolet .	1. Buy a new Chevrolet .
	They are already industrialized	They are already industrialized... .
	(Lavery vs. U.S. .)	(Lavery vs. U.S .)

The Jaccard similarity measure, known also as Jaccard index and Jaccard coefficient, focuses on the relative size of the intersection between the two sets with reference to their union. We consider a set of dependency arcs across the sentences in each annotation scheme as a sample set. We perform pairwise comparisons of the formats macro-averaging over all sentences in the dataset, e.g. computing total counts for the union and intersection for each pair of formats $\langle A, B \rangle$, using the formula:

$$J = \frac{|A \cap B|}{|A \cup B|} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

M_{11} represents the total number of dependency arcs present in two annotation schemes A and B ;

M_{01} represents the total number of dependency arcs present in the annotation scheme B but absent from the annotation scheme A ;

M_{10} represents the total number of dependency arcs present in the annotation scheme A but absent from the annotation scheme B ;

M_{00} represents the total number of possible dependency arcs absent from both annotation schemes.

$n = M_{11} + M_{01} + M_{10} + M_{00}$ is the total number of possible dependency arcs for the set of sentences.

The results on the 24 sentences from the PEST dataset are shown in Table 3.5 and on the 29,867 sentences of the WSJ sections 2-21 are presented in Table 3.6. We observe that the statistics on the small collection of WSJ sentences used as representative for complex phenomena (and even on the 10 sentences in Ivanova et al. (2012)) is showing the same trends as the numbers on the large corpus. This suggests that linguistic analysis of the exemplary constructions defines properties of the format on a large scale.

	CD	CP	SB	SD	EP	PA	PT	DT	DM
CD		.24	.488	.283	.189	.534	.241	.527	.132
CP	.24		.2	.184	.121	.187	.199	.167	.148
SB	.488	.2		.571	.145	.514	.298	.355	.157
SD	.283	.184	.571		.116	.334	.49	.251	.143
EP	.189	.121	.145	.116		.181	.164	.185	.446
PA	.534	.187	.514	.334	.181		.315	.471	.124
PT	.241	.199	.298	.49	.164	.315		.242	.132
DT	.527	.167	.355	.251	.185	.471	.242		.13
DM	.132	.148	.157	.143	.446	.124	.132	.13	

Table 3.5: Pairwise Jaccard similarity on 24 sentences from PEST. The highest similarity score for each dependency format is marked in bold font

	CD	CP	SB	SD	EP	PA	PT	DT	DM
CD		.23	.578	.384	.195	.625	.298	.584	.137
CP	.23		.241	.188	.18	.226	.222	.2	.13
SB	.578	.241		.72	.155	.538	.319	.389	.163
SD	.384	.188	.72		.19	.38	.466	.267	.136
EP	.195	.18	.155	.19		.21	.173	.219	.459
PA	.625	.226	.538	.38	.21		.334	.521	.141
PT	.298	.222	.319	.466	.173	.334		.273	.154
DT	.584	.2	.389	.267	.219	.521	.273		.144
DM	.137	.13	.163	.136	.459	.141	.154	.144	

Table 3.6: Jaccard similarity indexes averaged on 29,867 sentences from WSJ for all formats pairs. The highest similarity score for each dependency format is marked in bold font

F1 score is a weighted harmonic mean of the precision and recall:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Table 3.7 largely confirms the inferences that we can make from Jaccard values.

The similarities of the formats are comparatively low: Jaccard index does not exceed 72% for any pair and F1 score is bounded by 83%. The measures (unsurprisingly) show that the Stanford formats, SB and SD, are the most similar formats among all nine. This result is predictable, because SD can be obtained by an automatic conversion of SB. For the CoNLL representations the intercorrelation between the syntactic and semantic levels is low compared to Stanford formats. CD is most related to PA; and CP is very partial and hence is an outlier, it appears to

	CD	CP	SB	SD	EP	PA	PT	DT	DM
CD		.374	.732	.555	.326	.769	.459	.737	.242
CP	.374		.389	.316	.195	.368	.363	.333	.228
SB	.732	.389		.825	.269	.7	.484	.56	.28
SD	.555	.316	.825		.196	.551	.636	.422	.241
EP	.326	.195	.269	.196		.348	.295	.359	.643
PA	.769	.368	.7	.551	.348		.51	.685	.243
PT	.459	.363	.484	.636	.295	.51		.429	.259
DT	.737	.333	.56	.422	.359	.685	.429		.248
DM	.242	.228	.28	.241	.643	.243	.259	.248	

Table 3.7: F1 score as similarity indexes computed on 29867 sentences of WSJ for all formats pairs. The highest F1 score for each dependency format is marked in bold font

	CD	CP	SB	SD	EP	PA	PT	DT	DM
CD	20	1	12	5	7	16	6	16	3
CP	1	2	1	0	1	1	1	1	1
SB	12	1	20	11	4	9	3	10	4
SD	5	0	11	15	2	4	4	4	3
EP	7	1	4	2	21	7	6	7	8
PA	16	1	9	4	7	20	9	18	1
PT	6	1	3	4	6	9	16	9	3
DT	16	1	10	4	7	18	9	20	1
DM	3	1	4	3	8	1	3	1	13

Table 3.8: Pairwise unlabelled dependency overlap.

be most similar to SB. The DELPH-IN dependency representations demonstrate comparatively strong interoperability with other schemes, since DT corresponds well with CD syntactically, while DM correlates with EP among the more semantic formats. The DELPH-IN representations correlate with the Prague and Stanford schemes on the syntactic level, but do not exhibit similarity to any of these two on the semantic level. Interagreement between DT and DM is very low. The Prague syntactic level, PA, correlates well with CD, while the Prague semantic level, PT, is closer to SD; the interagreement between the Prague formats is fairly strong. EP, as we already observed, corresponds to DM and appears to be quantitatively remote from the other semantic schemes, such as CP, SD, PT.

Table 3.8 shows how many unlabeled dependency arcs each pair of formats have in common for our running example *A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice..* The values in the diagonal of the table show the total number of dependencies in a given representation.⁶ DT has the largest dependency arc overlap with PA and CD, and DM with EP.

Open et al. (2014) investigate in more detail commonalities and differences between the semantic dependencies of DM, EP and PT, and present some high-level statistics for these an-

⁶These numbers differ slightly from the ones reported in Ivanova et al. (2012) because we did further alignment of the formats, e.g. standardizing conversion of the DELPH-IN formats to the PTB-style tokenization and considering root nodes for all the formats in comparison.

		DM	EP	PT
(1)	# labels	51	42	68
(2)	% singletons	22.62	4.49	35.79
(3)	# edge density	0.96	1.02	0.99
(4)	% _g trees	2.35	1.30	56.58
(5)	% _g projective	3.05	1.71	53.29
(6)	% _g fragmented	6.71	0.23	0.56
(7)	% _n re-entrancies	27.35	29.40	9.27
(8)	% _g topless	0.28	0.02	0.00
(9)	# top nodes	0.9972	0.9998	1.1237
(10)	% _n non-top roots	44.71	55.92	4.36

Table 3.9: Statistics of the DM, EP and PT dependency graphs. %_n denotes non-singleton node percentages and %_g percentages over all graphs

notation schemes (see Table 3.9). EP has the most compact set of dependency labels (1), whilst PT is the most fine-grained. Singletons (2) are graphs consisting of a single isolated node with no edges, and correspond to tokens analyzed as semantically vacuous such as most punctuation marks in DM and PT and determiners in PT. In contrast to DM and PT, EP on average has more edges per non-singleton nodes (3) which likely reflects that there are nodes in the graph corresponding to functional words. PT appears to be more “tree-oriented” format than the other two, judging from the percentage of actual trees (4), the proportions of projective graphs (5), and the proportions of reentrant nodes (7). Small percentages of graphs without at least one top node (8) and of graphs with at least two non-singleton components that are not interconnected (6) can be viewed as indirect indicators of well-formedness. According to these criteria, DM exhibits more “warning flags” which may indicate imperfections in the DeepBank annotations or the conversion procedure from full MRS to bilexical dependencies, but possibly also exceptions to our intuitions about semantic dependency graphs. Oepen et al. (2014) also note that directionality of semantic dependencies is a major source of diversion between DM and EP on the one hand, and PT on the other hand.

3.6 Summary

In this section we presented an automatic conversion procedure of ERG annotations to bilexical dependencies. The work is motivated by the fact that the bilexical dependencies are useful in NLP applications, can be an attractive representation of the rich collection of the HPSG-annotated resources for a broader community and serve as a basis for the cross-framework parser comparison which we will discuss in the present thesis in Chapter 5. The conversion does not reduce HPSG to existing dependency schemes but derives new formats that embed core aspects of the syntactic analysis (DT) and basic predicate argument structure (DM) into the form of bilexical dependency graphs. This decision was made in order to protect the features of the original ERG annotations whereas the conversion to an existing scheme would require significant changes in the analysis of the linguistic phenomena and introduce noise for

the cases where there is no one-to-one correspondence. Finally, there is no unique standard in the field.

In order to analyze similarity of the newly developed syntactic and semantic dependency formats to the other existing schemes, we perform qualitative and quantitative analysis of a range of syntactico-semantic dependency formats. The comparison shows a large variation across formats and Jaccard similarity index is below 58% for any of the examined representation pairs. In the next section we will empirically compare the effects of the choice between three syntactic dependency annotation schemes on the accuracy of statistical parsers.

Chapter 4

Contrasting parsing experiments

In this chapter, we will inspect the correlation of the syntactic dependencies derived from ERG with existing schemes by analyzing the effects of the choice of syntactic dependency format on the performance of syntactic analyzers. For these goals, we carry out an experimental comparison of i) four syntactic dependency schemes—CoNLL Syntactic Dependencies (CD), Stanford Basic (SB), DELPH-IN Syntactic Derivation Tree (DT) and Prague dependencies of the analytical layer of representation (PA); ii) three “native” data-driven dependency parsers—Malt, MST and B&N; and iii) the influence of two different approaches to lexical category disambiguation (aka tagging) prior to parsing—PTB PoS tags and ERG lexical types. Comparing parsing accuracies in various setups, we study the interactions of these three aspects and analyze which configurations are easier to learn for a dependency parser. In addition, we contrast our syntactic dependency formats on the downstream task of negation resolution.¹

4.1 Related work

Dependency parsing is arguably one of the most active research areas in natural language processing in the past decade. As we recall from Chapter 3, dependency representations are useful for a number of NLP applications, for example, machine translation (Ding and Palmer, 2005), information extraction (Yakushiji et al., 2006), analysis of typologically diverse languages (Bunt et al., 2010) and parser stacking (Øvrelid et al., 2009). The CoNLL shared tasks 2006-2009 on dependency syntactic and semantic parsing (Buchholz and Marsi, 2006; Nivre et al., 2007a; Surdeanu et al., 2008; Hajič et al., 2009) caused an increased interest in parsing with dependency grammars and its applications. The important contribution of these competitions was a large-scale analysis of the state of the art in dependency parsing and extensive parser comparison, both intrinsic and in application to semantic role labeling (Surdeanu et al., 2008). As we saw in Chapter 2, the dominant approaches to parsing represented at the challenges were *transition-based* and *graph-based* frameworks. In terms of resources, important outcomes of the shared tasks were multilingual dependency corpora, the CoNLL data representation format, standardization of evaluation metrics and evaluation software, as well as the development of the CD and CP dependency schemes for English.

¹Some preliminary parsing results of this chapter are published in Ivanova et al. (2013b), and extrinsic evaluation results are published in Ivanova et al. (2015).

Dataset	# of sentences	average sent. length	agreement
WSJ	1796 (75%)	22.25	71.33
BROWN	375 (88%)	15.74	80.04
PCHEMTB	147 (75%)	23.99	69.27
CHILDES	595 (89%)	7.49	73.91

Table 4.1: Agreement between HPSG dependency backbone and CD in unlabeled attachment score (Zhang and Wang, 2009)

The Malt and MST parsers were the best performing parsers in the CoNLL-X 2006 shared task with notably close scores and the systems based on these parsers were also among the best ones in the CoNLL 2008 and 2009 shared tasks. In the open challenge at the CoNLL 2008 shared task, the system that performed best on syntactic dependencies was a hybrid architecture based on both Malt and MST parsers (Zhang et al., 2008), while in the closed challenge the second-best system incorporated MST (Che et al., 2008) and the third-best system relied on Malt (Ciaramita et al., 2008). In the open challenge of the multilingual CoNLL 2009 shared task the best system for English exploited MST for syntactic dependency parsing (Zhao et al., 2009) and in the closed challenge the best system for English, ranked second in the average evaluation for all the languages in the competition, is an extended version of MST-based architecture of Che et al. (2008) described in Che et al. (2009). One of the participating semantic role labeling systems in the CoNLL 2009 shared task exploited an HPSG dependency backbone derived from the ERG derivation tree (Zhang et al., 2009b) though only for extracting features for the MST parser.

Our DT format is obtained in a similar manner as the HPSG dependency backbone in the work of Zhang et al. (2009b) and Zhang and Wang (2009). Table 4.1 presents agreement in unlabeled attachment score between the HPSG dependency backbone and CD from Zhang and Wang (2009). We show their results for a similar parser setup that we use in the present thesis: only the sentences for which PET produced a complete analysis in a full parsing mode without fallback strategy are considered. With a perfect disambiguation model (manual parse selection from the parse forest produced by PET) results are improved by 8% compared to the 71.33% for WSJ with an automatic disambiguation.

Johansson and Nugues (2007) evaluate intrinsically and extrinsically the two dependency schemes obtained automatically from PTB using their LTH Constituent-to-Dependency Conversion Tool for Penn-style Treebanks and the Penn2Malt converter (Nivre, 2006). Compared to the former format, the latter scheme is non-projective and it has the richer set of arc labels. Intrinsic analysis of parsing the two dependency formats with Malt and MST shows that the structurally more complex format produced with the converter of Johansson and Nugues (2007) is more difficult for the parsers. In the extrinsic evaluation the authors study the impact of the annotation schemes on the task of semantic role classification. The accuracy of the semantic role classifier is 23% higher on the richer format derived from PTB with the converter of Johansson and Nugues (2007). A more granular set of arc labels that for example distinguishes between direct and indirect objects helps to improve the performance of the classifier.

Schwartz et al. (2012) present an empirical study of annotation scheme properties and how

these influence parser results. Six constructions, for which different formalisms often disagree, were selected for the analysis. For such syntactic structures the head criteria, mentioned in Section 2.1.2 of the present work, are either inapplicable (e.g. in a complex noun phrase “Heinz Holliger” neither of the components determines the syntactic or semantic category of the whole phrase and neither (or arguably both) restricts the form of the other) or in conflict (e.g. in a verb chain “would allow” the main verb bears the semantic category, but the modal verb defines that the main verb must be in the infinitive form). The authors investigate which dependency representations of several syntactic structures are easier to parse with several parsers, including Malt and MST. The results imply that all parsers consistently perform better when 1) coordination has one of the conjuncts as the head rather than the coordinating conjunction; 2) the noun phrase is headed by the noun rather than by the determiner; 3) the preposition/subordinating conjunction rather than the NP/clause serve as the head. Applying these predictions to the qualitative structural analysis carried out in Section 3.5.2, we can expect 1) Malt and MST to have fewer errors on coordination structures parsing SB and CD than parsing DT because SB and CD choose the first conjunct as the head and DT chooses the coordinating conjunction as the head; 2,3) no significant differences for the errors on the noun and prepositional phrases because all three schemes have the noun as the head of the noun phrase and the preposition as the head of the prepositional phrase. In addition, Schwartz et al. (2012) note a trend towards one of the annotations in complex verb groups and complex nouns structures: in most setups, the dependency format is easier to learn with the MST parser if the leftmost noun in complex noun sequences and the modal verb in verb groups are chosen as heads. From the analysis in Section 3.5.2, we would expect MST to make more mistakes in complex verb groups for SB than CD and DT, because the former suggests that the main verb is the head, while the latter two select the auxiliary or modal verb. With respect to the noun sequences, these three formats do not generally select the leftmost noun as the head.

Schwartz et al. (2012) view parsing purely as an engineering problem and suggest that an annotation scheme should have properties that are easier to process. Although it is very useful to know what is easier to work with empirically, eventually we are unlikely to modify a linguistic theory in order to improve performance of some software but would rather prefer to develop tools that can model the theory. Schwartz et al. (2012) present only intrinsic evaluation and leave the question open whether the proposed properties of an annotation scheme would be beneficial for downstream applications.

Miwa et al. (2010) present intrinsic and extrinsic evaluation of several constituency and grammar-based parsers on the CD format and several versions of Stanford dependencies. The target representations are obtained via standard conversions from PTB-style trees as described in Section 2. The authors make the interesting observation that a higher dependency performance does not always lead to better performance of a downstream application that relies on the parser. For example, it could be more beneficial to use the Stanford dependencies rather than CD in the event extraction system although in the intrinsic evaluation most parsers predicted the types and relations of the CD scheme more accurately than those of the Stanford representation. With respect to the results of this work, CD appears to be a slightly easier representation to learn than Stanford though we have to keep in mind that in our work we use different types of parsers and a different dataset.

Cer et al. (2010) contrast a number of parsers in terms of accuracy and speed on the task

	Attachment F_1 , %		Time, min:sec	
	Unlabeled	Labeled	Parsing	Total
Malt (Covington)	80.0	76.6	0:09	0:16
Malt (Nivre Eager)	80.1	76.2	0:08	0:16
Malt (Nivre)	80.2	76.3	0:08	0:15
Malt (Nivre Eager with feature interactions)	84.8	81.1	3:15	3:23
MST (Eisner)	82.6	78.8	5:54	6:01

Table 4.2: Unlabeled and labeled attachment F_1 score and time to generate Stanford Standard dependencies with the Malt and MST parsers (Cer et al., 2010)

of producing Stanford dependencies on the section 22 of PTB. Even though the comparison is performed on the standard (collapsed and propagated) version of the Stanford scheme, the dependency parsers are trained on SB like in the experiments in the present thesis and their outputs are transformed to the SD representation. The aspect of the work of Cer et al. (2010) related to our experiments concern the comparison of Malt and MST, see Table 4.2. The former parser is the fastest in the comparison, e.g. the total time for PoS-tagging, parsing, dependency extraction and conversion to SD does not exceed 16 seconds for the Covington, Nivre Eager and Nivre algorithms included in the Malt package, while the corresponding time for the MST parser with the Eisner algorithm is 6 minutes. However, MST produces more accurate results than Malt with the three above mentioned fast algorithms. The error analysis suggests that, unsurprisingly, parsers make mistakes on structures that are known to be hard to attach: subordinate clauses, prepositional and adverbial phrases. The Malt parser with the Covington, Nivre Eager and Nivre algorithms tends to produce more local attachments than the MST parser, though erroneous local attachments are also among the mistakes that the MST parser makes due to its feature set which favors short distance dependencies. One of the conclusions of the article is that the Malt package is an optimal choice for parsing large corpora to SD when speed is the most important.

Bender et al. (2011) perform construction-based evaluation of several parsers on the linguistic phenomena shown in Table 4.3. We are primarily interested in their results for the MST parser trained on PTB in the CD annotation format. The evaluation was performed on the English Wikipedia data against manual annotations. Table 4.4 summarizes individual recalls of MST for different linguistic phenomena in the experiments. The parser performed better on comparatively well-studied constructions such as control and verbal gerunds and for a long-distant dependency in tough adjectives but had lower recall for the other phenomena even though most of them involve local dependencies.

The EVALITA Dependency Parsing Tasks of 2007, 2009 and 2011 (Bosco and Mazzei, 2011) aimed to compare parsing paradigms and annotation formats for the Italian language. As in the CoNLL shared tasks, the data for EVALITA tasks is represented in CoNLL format and the main evaluation metrics are LAS and UAS. The parsers that were competing in the tasks represent three frameworks: transition-based, graph-based and rule-based parsing. The reference resource for the tasks is the Turin University Treebank (TUT) annotated in the theoretical framework of Word Grammar (Hudson, 1990) which is reflected in the choice of lexical heads in noun and prepositional phrases and complex verbs structures, and the annotation format adheres to the projectivity constraint. The dependency format for the task is derived

Phenomena	Example
Control	<i>Alfred</i> “retired” in 1957 at age 60 but continued to paint full time
Verbal gerunds	Accessing <i>the website</i> without the “www” subdomain returned a copy of the main site for “EP.net”.
Tough adjectives	Original <i>copies</i> are very hard to find .
Interleaved arg/adj	The story shows , <i>through</i> flashbacks, the different <i>histories</i> of the characters.
Right node raising	Ilúvatar, as his names imply, exists before and independently of <i>all else</i> .
Verb + particle	He once threw out two <i>baserunners</i> at home in the same inning.
Bare relatives	This is the <i>second time</i> in a row Australia had lost their home tri-nations’ series.
Absolutives	The format consisted of 12 games, each <i>team</i> facing the other teams twice.
Adj/Noun2 + Noun1- <i>ed</i>	<i>Light colored glazes</i> also have softening effects when painted over dark or bright images.
Expletive <i>it</i>	Crew negligence is blamed, and <i>it</i> is suggested that the flight crew were drunk.

Table 4.3: Example sentences for various linguistic phenomena from Bender et al. (2011)

Phenomena	MST recall	Preferred head	Preferred dependent
Control	92%	“upstairs” verb	“downstairs” verb
Verbal gerunds	85%	selecting head	gerund
Tough adjectives	85%	tough adjective	<i>to</i> -VP complement
Interleaved arg/adj	81%	selecting verb	interleaved adjunct
Right node raising	78%	verb/prep2	shared noun
Verb + particle	76%	particle	complement
Bare relatives	71%	gapped predicate in relative	modified noun
Absolutives	70%	absolutive predicate	subject of absolutive
Adj/Noun2 + Noun1- <i>ed</i>	66%	head noun	Noun1- <i>ed</i>
Expletive <i>it</i>	30%	<i>it</i> -subject taking verb	<i>it</i>

Table 4.4: Recall of MST on 10 linguistic phenonema and the preferred choices of head and dependent (Bender et al., 2011)

automatically from the corpus annotation. Relation labels of the native TUT annotation can include three components, i.e. morpho-syntactic, functional-syntactic and syntactic-semantic (e.g. VERB-INDCOMPL-LOC) while the dependency format used to train parsers exploits only the functional syntactic component (e.g. INDCOMPL). Similarly, in our experiments we reduce the rich set of ERG constructions to a smaller set of more general dependency labels (e.g. *sb-hd_mc_c* is reduced to *sb-hd*), see below. In EVALITA 2009 TUT dependencies are contrasted to the annotation scheme produced semi-automatically from the ISST corpus (Montemagni et al., 2003) by combining information from different annotation levels of the corpus. The head selection is motivated by both syntactic and semantic criteria, e.g. in the determiner-noun construction the head is the noun (semantic choice) while in the preposition-noun case the head is the preposition (syntactic choice). The annotation scheme permits non-projective dependencies. Bosco et al. (2010) compare the influence of the two dependency formats derived from TUT and ISST on dependency parsing. The structures that are hard to parse with Malt and other parsers in the comparison require semantic interpretation, e.g. appositions, unrestrictive modifiers and indirect object for the first dependency scheme and locative, temporal and indirect complements for the second dependency format. Coordination and punctuation are also challenging for automatic processing. Among the dependencies for which parsers performed best are local relations. Bosco et al. (2010) conclude that some distinctions encoded in a dependency representation are difficult to generalize for statistical parsers. However, we should note

that, unlike in our experiments, the datasets in this work differ not only in the annotation styles but these are two different collections of texts representing different genres which also had a certain impact on the parsing performance.

Candito et al. (2010) compare parsers, including Malt and MST, in terms of labeled and unlabeled accuracy and parsing times for French. Similarly to the PTB for English used in our work, the native format of the French Treebank is constituency-based and dependencies are obtained via conversion using head propagation rules. Candito et al. (2010) observe that the differences in performance between the parsing architectures are small. The best overall labeled accuracy was achieved with the MST parser supplied with predicted PoS, lemmas, morphological features and clusters of word forms.

The Bohnet and Nivre (2012) parser (dubbed B&N), described in detail in Section 2.2.1, combines the transition-based approach with global modeling. Unlike most dependency parsers that presuppose morphological pre-processing of the input words of the sentence, this parser performs tagging jointly with parsing. The software has been compared to the state-of-the-art statistical dependency analyzers on WSJ data and it has given a very competitive result of 93.38% UAS outperforming Malt and MST (Bohnet and Nivre, 2012).

In this overview of related work we have seen the important role of transition-based and graph-based approaches to dependency parsing in the last few years which motivated our choice of parsers. Previous work on parser comparison indicates that the Malt and MST parsers have strong positions in the dependency parsing universe and the previous analyses of parser outputs give indications about which particular linguistic phenomena are easier and which are especially difficult to process with those two parsers. It has also been shown that design choices in the dependency annotation format may have a significant impact on parser performance and it is therefore interesting to further investigate the effects of the choice of dependency formalism on parsing accuracy.

4.2 Experimental setup

In the following we will describe the experimental setup for the experiments performed in this chapter. We will briefly report on dependency formats, tokenization approaches, PoS tags and dependency labels, data sets, parsers and PoS taggers.

Dependency schemes

In this work we extract DeepBank data in the form of bilexical syntactic dependencies, the *DELPH-IN Syntactic Derivation Tree (DT)* format, described in Section 3.4.1. We obtain the exact same sentences in *Stanford Basic (SB)* format from the PTB with the Stanford converter, in the *CoNLL Syntactic Dependencies (CD)* representation from the PTB with the LTH Constituent-to-Dependency Conversion Tool for Penn-style Treebanks and in *Prague dependency format of the analytical layer (PA)* from the English subpart of the Prague Czech-English Dependency Treebank (PCEDT) (Čmejrek et al., 2004) with the Treex system (Popel and Žabokrtský, 2010) as detailed in Section 3.5.1.

SB, CD and PA represent direct conversions of PTB analyses to bilexical dependencies; in contrast, DT is grounded in the linguistic theory of HPSG and captures decisions taken in the



Figure 4.1: Annotation of coordination structure in SB, CD, DT and PA dependency formats

grammar such as directionality of the dependencies (for modifier-head vs. head-modifier constructions), the dependency names and the supertags. Figure 4.1 demonstrates the differences between the formats on the coordination structure. According to Schwartz et al. (2012), the chosen analysis of coordination in SB and CD should be easier for a statistical parser to learn; however, as we will see in Section 4.4, the DT analysis has more expressive power distinguishing structural ambiguities illustrated by the classic example of local vs. shared modification in coordinate structures *old men and women*.

Tokenization alignment for DT

Both the SB and CD dependency schemes are automatically derived from PTB and therefore support PTB-style tokenization, which means for comparative parsing experiments we have to align DT so that the tokens are exactly the same in all three schemes. As we already mentioned in Section 3.4.3, PTB-style tokenization is produced at the early stages of analysis with ERG, but the full HPSG derivation tree is available only with the ERG-style tokenization, and therefore needs some accommodations during the transformation to bilexical dependencies. Below we describe which modifications are necessary to obtain DT dependencies with PTB-style tokenization.

Punctuation Punctuation is attached to words in the ERG derivation tree and has to be split for the alignment with PTB, as illustrated in Example (59). The new dependency link has the name “PUNCT”.



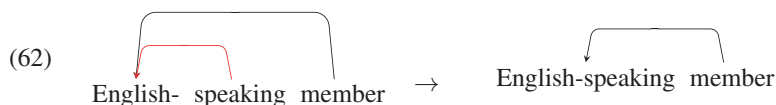
Multiword lexical items Multiword lexical items (*North America*, *according to*) are represented as one token in the ERG derivation tree. The right-most non-punctuation token is considered the head and all the remaining tokens become dependents on this head with the label “MWE”, see Example (60).



Negation The contracted negative form of a verb acts as one leaf node in the derivation tree, and when splitting it into component parts we establish a dependency from the verb to the negation part with a dependency label “NEG”, as shown in Example (61).



Hyphenated words The component parts of hyphenated words are individual tokens in a derivation tree, therefore they simply have to be merged into one token for alignment with the data tokenized in the PTB-style, as demonstrated in Example (62).



PoS tags and dependency labels for DT

In the DT scheme, HPSG constructions are used as dependency labels and ERG lexical types are used as PoS tags. Our first experiment concerns the choice of the level of detail conveyed by the PoS tags and dependency labels. We use MaltParser 1.6.1 and the WeScience resource with sections 1-11 as a training set and section 13 as a test set (see Table 4.5). As we can see from Table 4.6, the size of the inventory of PoS tags varies dramatically depending on our choice of generalization.

WS01	WS02	WS03	WS04	WS05	WS06	WS07	WS08	WS09	WS10	WS11	WS12	WS13
615	816	744	776	664	710	655	721	809	725	599	628	810
7834												
9272												

Table 4.5: Number of sentences in the WeScience corpus

PoS type	# possible tags	# tags seen on training set
cut on 1st “_” (e.g. n)	11	11
cut on 2nd “_” (e.g. n_pp)	101	82
cut on 3rd “_” (e.g. n_pp_c-of)	971	663
complete (e.g. n_pp_c-of_le)	971	664

Table 4.6: The number of different PoS tags and the number of different PoS tags seen on the training data

Table 4.7 shows some observations from our experiments that investigate different granularities of PoS tags and dependency labels. Our conclusions are the following:

PoS type	ERG-style tok.						PTB-style tok.		
	Dep. labl. cut on 1st “_”			Dep. labl. complete			Dep. labl. cut on 1st “_”		
	LAS	UAS	LACC	LAS	UAS	LACC	LAS	UAS	LACC
cut on 1st “_”	79.75	83.25	83.64	77.02	83.59	80.34	81.90	85.82	85.27
cut on 2nd “_”	82.13	84.64	85.83	79.58	84.62	82.64	84.56	87.12	87.76
cut on 3rd “_”	84.11	85.56	88.00	81.91	85.37	85.12	85.32	86.79	88.78
complete	84.15	85.55	88.00	81.99	85.42	85.21	85.62	87.11	88.96

Table 4.7: Evaluation script: eval.pl. MaltParser results on WeScience corpus with derivation tree representation converted to bilexical dependencies with dependency label cut on the first “_” symbol and preserving ERG tokenization with variations in the choice of PoS tags

- The accuracy is the best when we cut the dependency label on the first “_”
- The accuracy is the best when we use the full ERG PoS tag rather than simplified ones (some results were better for the ERG PoS tag cut before the third “_”, but the statistical significance test showed that the difference is not statistically significant)
- When trained on short PoS tags, PTB-style tokenization is significantly better. When trained on long PoS tags, there is no significant difference between the two tokenization styles.

Treebanks

For the experiments in this chapter we use the *Penn Treebank* (Marcus et al., 1993) and the *DeepBank* (Flickinger et al., 2012) resources. As we recall from Section 2.5, DeepBank 1.0 is comprised of roughly 85% of the sentences of the first 22 sections of the Penn Treebank annotated with full HPSG analyses from the English Resource Grammar (ERG). The DeepBank annotations are created on top of the raw text of PTB. Due to tokenization errors in the PTB (which DeepBank corrects) as well as imperfections of the automatic tokenization in the ERG parser (which introduce new tokenization errors), there are some token mismatches between DeepBank and PTB. We had to filter out such sentences to have a consistent number of tokens in the DT, SB and CD formats. For our experiments we use sections 0-19 of DeepBank comprising 33,334 sentences as a training set, section 20 consisting of 1700 sentences as a development set and section 21 comprising 1389 sentences as a test set.

Parsers

We use parsers that adopt different approaches and implement various algorithms. These were described in detail in Chapter 2, but we briefly repeat some of their defining properties below and provide relevant experimental settings for each parser.

Malt: transition-based dependency parser with local learning and greedy search. We exploited version 1.7.2 and performed automatic tuning with MaltOptimizer-1.0.2. For half of the configurations MaltOptimizer chose the Nivre Standard algorithm as optimal, and Projective

Stack for the other half. Both algorithms use essentially the same transitions and are limited to projective dependency trees.

MST: graph-based dependency parser with global near-exhaustive search. We used the MST parser version 0.5.0 with the projective Eisner algorithm with second order features (i.e. features over pairs of adjacent edges in the tree), including punctuation in the hamming loss calculation, creating the training forest and using the 1-best parse set size constraint during training.

B&N: transition-based dependency parser with joint tagger that implements global learning and beam search. We set the beam parameter to 80 and otherwise employed the default setup of version 3.3 of the parser.

As we have already seen, the Malt and MST parsers represent two prominent paradigms for learning and inference: transition-based and graph-based parsing and B&N combines these approaches. Both Malt and MST require pre-tagged inputs.

Part-of-speech tags

We experiment with two tag sets: the *PTB tags* and the lexical types of the ERG, the so-called *supertags*. PTB tags determine *part-of-speech* and *morphological features* such as number for nouns, degree of comparison for adjectives and adverbs, tense and agreement with person and number of subject for verbs etc. Some of the PTB tags assign a syntactic function, for example in the case of uninflected verbs there is a distinction between infinitive or imperative (VB) and non-third person singular present tense (VBP).

As we recall from Section 2.2.2, supertags are composed of *part-of-speech*, *valency* in the form of an ordered sequence of complements, and *annotations* that encompass category-internal subdivisions, e.g. mass vs. count vs. proper nouns, intersective vs. scopal adverbs, referential vs. expletive pronouns. Example of a supertag: *v_np_is_le* (verb “is” that takes noun phrase as a complement).

The two tag sets are of very different size initially: there are 48 tags in the PTB tag set and 1091 supertags in the set of lexical types of the ERG. The state-of-the-art accuracy of PoS-tagging on in-domain test data using gold-standard tokenization is roughly 97% for the PTB tag set and approximately 95% for the ERG supertags (Ytrestøl, 2011).

Predicted PoS tags

Since parsing with the gold standard PoS tags is an idealized scenario, we prepare the automatically tagged inputs. We generate supertags using the *übertagger* of Dridan (2013a). The *übertagger* starts from raw strings and provides an output in the ERG-style tokenization reproducing the gold tokenization with an accuracy of about 93%. Therefore the output needs conversion to PTB-style tokenization, which is very similar to the conversion of the ERG derivation trees to bilexical dependencies in PTB-style tokenization discussed in Section 3.4.3. The results of the evaluation of the *übertagger* performance before the conversion to PTB tokenization are summarized in Table 4.8 and the results after the conversion are shown in Table 4.10.

For the automatic PTB PoS tagging, we experiment with the English Stanford Tagger version 3.4 (Manning, 2011) and the TnT tagger (Brants, 2000). PTB tags in the gold DeepBank are predicted with the TnT tagger, but from Table 4.9 the Stanford tagger appears to perform

	Test, supertags Übertagger
Tag precision	91.21
Tag recall	90.84
Tag F1-score	91.02

Table 4.8: Evaluation of automatic übertagging on the test set before conversion to PTB tokenization (using the DeepBank sections 0-19 for training and the DeepBank section 21 for testing)

	Dev, PTB tags	
	Stanford Tagger	TnT
Total # tags right	36547	36363
Total # tags wrong	1098	1282
Tag recall, %	97.08	96.59

Table 4.9: Performance of the English Stanford Tagger and TnT taggers on the development set (using the DeepBank sections 0-19 for training and the DeepBank section 20 for testing)

	Test, PTB tags		Test, supertags	
	Stanford Tagger	B&N, DT	Übertagger	B&N, DT
Total # tags right	29918	29873	28422	28190
Total # tags wrong	893	938	2389	2621
Tag recall, %	97.10	96.96	92.25	91.49

Table 4.10: Results of the automatic PTB PoS tagging with the English Stanford Tagger and the B&N parser and the automatic supertagging with the übertagger of Dridan (2013a) and the B&N parser on the test set (using the DeepBank sections 0-19 for training and DeepBank section 21 for testing)

slightly better therefore it was chosen for further experiments. Table 4.10 shows the performance of the Stanford tagger on the test set.

The B&N parser starts from the raw strings and predicts PoS tags during parsing. Its tagging performance with PTB tags and supertags on the test set is presented in Table 4.10.

Evaluation

As our main evaluation metrics, we chose three measures that were used in CoNLL shared tasks: labeled accuracy score (LAS), unlabeled accuracy score (UAS) and label accuracy (LACC). To compute these metrics, we run the evaluation script `eval.pl` which was developed and standardized for these competitions. To determine whether results are statistically significant we used Dan Bikel’s Randomized Parsing Evaluation Comparator (Buchholz and Marsi, 2006) at the 0.001 significance level.

Malt parser, w/o punctuation						
<i>PTB tags</i>						
	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	89.89	88.16	91.45	90.71	93.91	92.44
CD	89.07	87.74	92.41	91.55	91.34	90.26
DT	87.54	86.76	89.80	89.22	89.88	89.22
PA	81.53	79.79	84.81	83.82	90.29	88.78

<i>Supertags</i>						
	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	88.80	85.88	91.16	88.81	92.98	90.84
CD	88.80	86.51	91.86	90.13	91.64	89.69
DT	90.65	84.07	91.54	87.05	93.05	87.19
PA	80.07	77.69	84.24	82.37	88.72	87.12

<i>PTB tags + supertags</i>						
	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	91.04 ¹	88.32 ¹	92.48 ¹	90.77 ¹	94.73 ¹	92.59 ¹
CD	90.25 ¹	87.94 ¹	93.22 ¹	91.44 ¹	92.57 ¹	90.60 ¹
DT	91.68 ¹	85.85 ¹	92.54 ¹	88.82 ¹	93.70 ¹	88.29 ¹
PA	82.51 ¹	79.82 ¹	85.59 ¹	83.80 ¹	90.86 ¹	88.89 ¹

Table 4.11: Parsing results of Malt on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. Punctuation is excluded from the scoring. *PTB tags*: Malt is trained and tested on PTB tags. *Supertags*: Malt is trained and tested on supertags. *PTB tags + supertags*: Malt is trained on PTB tags and supertags. ¹ denotes a feature model in which PTB tags function as PoS and supertags act as additional features (in CPOSTAG field)

4.3 Results

We will now go on to present and discuss the results obtained in our experiments and examine these in terms of the parser employed, the chosen dependency format and PoS tag set. Tables 4.11, 4.12 and 4.13 present scores ignoring punctuation and the corresponding values in Tables 4.14, 4.15 and 4.16 present scores including punctuation.

From the *parser* perspective, MST is better than Malt for attachment in the traditional setup with gold PTB tags on all the formats; on CD and DT MST has higher labeled and unlabeled accuracy than Malt, and this picture is preserved when PTB tags are predicted with the Stanford tagger (Table 4.11 and 4.12, *PTB tags*). The B&N parser outperforms both Malt and MST on the CD, DT and PA formats in terms of labeled dependency score even though it does not receive gold PTB tags during the test phase but predicts them (Table 4.13, *Predicted PTB tags*). This can possibly be explained by the fact that B&N implements a novel approach to parsing: beam-search algorithm with global structure learning.

MST parser, w/o punctuation*PTB tags*

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	90.00	88.10	92.38	91.35	93.75	92.11
CD	89.92	88.63	93.40	92.49	92.25	91.19
DT	89.41	88.47	91.98	91.14	91.57	90.87
PA	82.00	80.21	86.88	85.74	89.74	88.32

Supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	87.21	84.22	90.28	87.85	91.67	89.52
CD	88.84	86.34	91.98	90.20	91.84	89.77
DT	91.33	85.38	92.11	88.32	93.86	88.63
PA	79.30	77.09	84.83	83.08	87.81	86.42

PTB tags + supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	90.65 ¹	88.38 ¹	93.20 ¹	91.76 ¹	94.02 ¹	92.30 ¹
CD	90.26 ¹	88.60 ¹	93.86 ¹	92.56 ¹	92.55 ²	91.20 ¹
DT	93.23 ²	88.14 ¹	94.17 ¹	90.98 ¹	95.12 ²	90.52 ¹
PA	82.48 ¹	80.12 ¹	87.48 ¹	85.69 ¹	89.89 ¹	88.30 ¹

Table 4.12: Parsing results of MST on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. Punctuation is excluded from the scoring. *PTB tags*: MST is trained and tested on PTB tags. *Supertags*: MST is trained and tested on supertags. *PTB tags + supertags*: MST is trained on PTB tags and supertags. ¹ denotes a feature model in which PTB tags function as PoS and supertags act as additional features (in CPOSTAG field); ² stands for the feature model which exploits gold supertags as PoS and uses PTB tags as extra features (in CPOSTAG field)

B&N, w/o punctuation						
<i>Predicted PTB tags</i>						
	LAS		UAS		LACC	
SB	90.65		93.05		94.01	
CD	90.92		93.97		92.94	
DT	90.91		93.18		92.76	
PA	83.65		87.15		91.02	
<i>Predicted supertags</i>						
	LAS		UAS		LACC	
SB	86.88		90.59		91.02	
CD	87.97		92.05		90.55	
DT	87.56		90.39		90.09	
PA	79.91		84.72		88.19	
<i>Predicted PTB tags + gold/predicted supertags</i>						
	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	91.57	91.29	93.81	93.59	94.62	94.44
CD	91.32	91.11	94.24	94.11	93.28	93.15
DT	93.37	90.19	94.34	92.85	95.07	92.08
PA	83.96	83.31	87.40	86.96	91.18	90.74
<i>Predicted supertags + gold/predicted PTB tags</i>						
	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	88.48	87.73	91.15	91.09	92.47	91.76
CD	88.92	88.51	92.47	92.30	91.37	90.96
DT	87.93	87.86	90.75	90.73	90.36	90.32
PA	81.35	80.75	85.64	85.31	89.35	88.86

Table 4.13: Parsing results of B&N on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. The parser is trained on gold-standard data. Punctuation is excluded from the scoring. *Predicted PTB*: the parser predicts PTB tags during the test phase. *Predicted supertags*: the parser predicts supertags during the test phase. *Predicted PTB + supertags*: the parser receives supertags as feature and predicts PTB tags during the test phase. *Predicted supertags + PTB*: the parser receives PTB tags as feature and predicts supertags during the test phase

Malt parser, with punctuation*PTB tags*

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	88.81	87.16	90.19	89.40	94.57	93.27
CD	88.30	87.12	91.24	90.46	92.32	91.38
DT	89.03	88.35	91.02	90.52	91.10	90.53
PA	80.43	78.76	83.74	82.73	90.39	89.05

Supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	88.01	85.07	90.08	87.65	93.80	91.92
CD	87.96	85.67	90.66	88.84	92.58	90.87
DT	91.78	85.96	92.56	88.57	93.90	88.71
PA	78.86	76.52	82.89	81.03	88.93	87.54

PTB tags + supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	90.09 ¹	87.44 ¹	91.36 ¹	89.61 ¹	95.34 ¹	93.02 ¹
CD	89.50 ¹	87.26 ¹	92.13 ¹	90.34 ¹	93.54 ²	91.66 ¹
DT	92.68 ¹	87.52 ¹	93.44 ¹	90.13 ¹	94.47 ¹	89.67 ¹
PA	81.33 ¹	78.76 ¹	84.44 ¹	82.64 ¹	90.91 ¹	89.18 ¹

Table 4.14: Parsing results of Malt on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. Punctuation is included in the scoring. *PTB tags*: Malt is trained and tested on PTB tags. *Supertags*: Malt is trained and tested on supertags. *PTB tags + supertags*: Malt is trained on PTB tags and supertags. ¹ denotes a feature model in which PTB tags function as PoS and supertags act as additional features (in CPOSTAG field); ² stands for the feature model which exploits supertags as PoS and uses PTB tags as extra features (in CPOSTAG field)

MST parser, with punctuation*PTB tags*

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	89.37	87.53	89.37	90.40	94.50	93.07
CD	89.45	88.20	92.51	91.60	93.16	92.23
DT	90.64	89.82	92.91	92.18	92.56	91.95
PA	81.19	79.49	85.95	84.82	89.95	88.69

Supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	86.44	83.58	89.13	86.77	92.69	90.80
CD	88.13	85.75	90.90	89.14	92.81	91.00
DT	92.38	87.10	93.06	89.67	94.62	89.98
PA	78.40	76.31	83.71	82.05	88.25	87.02

PTB tags + supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	89.98 ¹	87.82 ¹	92.23 ¹	90.79 ¹	94.74 ¹	93.23 ¹
CD	89.78 ¹	88.20 ¹	92.96 ¹	91.69 ¹	93.42 ²	92.23 ¹
DT	94.04 ²	89.51 ¹	94.85 ²	92.01 ¹	95.72 ²	91.62 ¹
PA	81.68 ¹	79.47 ¹	86.54 ¹	84.84 ¹	90.08 ¹	88.67 ¹

Table 4.15: Parsing results of MST on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. Punctuation is excluded from the scoring. *PTB tags*: MST is trained and tested on PTB tags. *Supertags*: MST is trained and tested on supertags. *PTB tags + supertags*: MST is trained on PTB tags and supertags. ¹ denotes a feature model in which PTB tags function as PoS and supertags act as additional features (in CPOSTAG field); ² stands for the feature model which exploits gold supertags as PoS and uses PTB tags as extra features (in CPOSTAG field)

B&N, with punctuation*Predicted PTB tags*

	LAS	UAS	LACC
SB	90.26	92.38	94.72
CD	90.57	93.26	93.75
DT	91.98	93.98	93.62
PA	82.91	86.37	91.13

Predicted supertags

	LAS	UAS	LACC
SB	86.47	89.74	92.12
CD	87.56	91.16	91.66
DT	88.99	91.48	91.26
PA	79.19	83.80	88.62

Predicted PTB tags + gold/predicted supertags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	91.16	90.90	93.14	92.92	95.26	95.10
CD	90.87	90.68	93.44	93.32	94.05	93.94
DT	94.15	91.31	95.01	93.65	95.65	92.99
PA	83.24	82.63	86.63	86.21	91.29	90.89

Predicted supertags + gold/predicted PTB tags

	LAS		UAS		LACC	
	gold	pred.	gold	pred.	gold	pred.
SB	87.96	87.30	90.31	90.25	93.37	92.75
CD	88.36	88.02	91.50	91.37	92.35	91.99
DT	89.32	89.25	91.80	91.77	91.50	91.46
PA	80.55	80.02	84.72	84.42	89.61	89.17

Table 4.16: Parsing results of B&N on the Stanford Basic (SB), CoNLL Syntactic Dependencies (CD), DELPH-IN Syntactic Derivation Tree (DT) and Prague analytical layer (PA) formats. The parser is trained on gold-standard data. Punctuation is included in the scoring. *Predicted PTB*: the parser predicts PTB tags during the test phase. *Predicted supertags*: the parser predicts supertags during the test phase. *Predicted PTB + supertags*: the parser receives supertags as feature and predicts PTB tags during the test phase. *Predicted supertags + PTB*: the parser receives PTB tags as feature and predicts supertags during the test phase

	Malt Supertags		
	LAS	UAS	LACC
SB	87.44	89.92	91.95
CD	87.37	90.52	90.43
DT	87.86	89.53	90.63
PA	78.72	83.07	87.80
	PTB tags + supertags		
	LAS	UAS	LACC
SB	89.19 ¹	91.54 ¹	93.12 ¹
CD	88.66 ¹	92.08 ¹	91.28 ¹
DT	89.18 ¹	90.90 ¹	91.43 ¹
PA	80.43 ¹	84.24 ¹	89.20 ¹

Table 4.17: Performance of Malt on the data annotated with PET-predicted supertags. PTB tags are predicted with the Stanford tagger, supertags are predicted with the PET parser; punctuation is excluded from the scoring. *Supertags*: Malt is trained and tested on supertags. *PTB tags + supertags*: Malt is trained on PTB tags and supertags. ¹ denotes a feature model in which PTB tags function as PoS and supertags act as additional features (in CPOSTAG field)

MST results deteriorate more than Malt when parsing SB and PA with gold supertags as compared to PTB tags and the performance of Malt on these two formats is better than the performance of MST (Table 4.11 and 4.12, *Supertags*). As we recall, this parser exploits context features “PoS tag of each intervening word between head and dependent” (McDonald et al., 2006). Due to the far larger size of the supertag set compared to the PTB tag set, such features are sparse and have low frequencies. This leads to the lower scores of parsing accuracy for MST. However, the transition from gold PTB to gold supertags affects parsing accuracy of Malt and MST on DT to the better and MST outperforms Malt (compare Tables 4.11 and 4.12, *PTB tags*, and Tables 4.11 and 4.12, *Supertags*). With predicted supertags MST has a larger drop in labeled accuracy than Malt on all the formats and Malt has significantly better results than MST on SB and DT.

We find that the combination of gold PTB tags and gold supertags improves the performance of Malt on all the formats and MST on SB and DT, but the difference is not significant when the supertags are predicted by the supertagger (Tables 4.11 and 4.12, *PTB tags + supertags*). For B&N we also observe a rise of accuracy on SB and DT when gold supertags are provided as feature for prediction of PTB tags (compare Table 4.13, *Predicted PTB tags*, and Table 4.13, *Predicted PTB tags + gold supertags*). However, the combination of PTB tags with gold supertags is not beneficial for parsing CD and PA, since the improvements are not statistically significant.

In terms of efficiency, the parsers have different training times: it takes minutes to train Malt, a few hours to train MST and about a day or longer to train the B&N parser. As stated by Cer et al. (2010), Malt also has shorter running time therefore this parser is a good choice when the speed is critical.

From the point of view of the *dependency format*, parsers have the highest LACC on SB

and first-rate UAS on CD in most of the configurations (Tables 4.11, 4.12 and 4.13). This means that SB is easier to label and CD is easier to parse structurally. DT appears to be a more difficult target format for Malt and MST. This is not an unexpected result, since SB and CD are both derived from PTB phrase structure trees directly and are oriented towards the dependency parsing task. DT is not custom-designed for dependency parsing and is independent from parsing-related issues in this sense. Unlike SB and CD, it is linguistically informed by the underlying, full-fledged HPSG grammar. The CD format appears to be least affected by interchanging gold supertags with predicted supertags. DT is the most affected, and the labeled accuracy of Malt and MST on DT is significantly lower than on CD when supertags are used as PoS and punctuation is not scored. Since DT and supertags are derived from the same grammar, this result is explained by the fact that DT relies on the fine-grained type categorizations made in ERG.

It is interesting to observe that the labeled accuracy scores of B&N are similar or better for DT than SB and CD in most of the configurations. This result motivates our choice of DT as a base for head-to-head parser comparison in Chapter 5 since B&N is able to learn the DT scheme to the same level of performance as the SB and CD schemes. The surprising result is that PA appears to be a significantly harder annotation scheme than SB, CD and DT for statistical parsers as their LAS drop by more than 7% for PA. The latter format is harder both structurally and in labeling, though the difference with other schemes in LACC is less drastic than in LAS and UAS.

As we recall from Chapter 3, Jaccard similarity values show that CD and DT are structurally closer to each other than SB and DT. Contrary to our expectations, the accuracy scores of the parsers do not suggest that CD and DT are particularly similar to each other in terms of parsing.

Inspecting the aspect of *tag set* we conclude that traditional PTB tags are compatible with SB, CD and PA but do not fit the DT scheme equally well, while ERG supertags are specific to the ERG framework and do not seem to be appropriate for SB, CD and PA. Neither of these findings seem surprising, as PTB tags were developed as part of the treebank from which SB, CD and PA are derived; whereas ERG supertags are closely related to the HPSG syntactic structures captured in DT. PTB tags were designed to simplify PoS-tagging whereas supertags were developed to capture information that is required to analyze syntax in terms of HPSG analyses.

For B&N the complexity of supertag prediction has significant negative influence on the attachment and labeling accuracies (compare *Predicted PTB tags* and *Predicted supertags* in Table 4.13). The addition of gold PTB tags as a feature lifts the performance of B&N and its accuracy is even higher than of MST on SB, CD and PA with gold supertags (compare Table 4.13, *Predicted supertags*, and Tables 4.11, 4.12, *supertags*).

With the PET parser we are able to predict supertags with a recall of 95.01% which is significantly more accurate than 92.25% with the standalone *übertagger*. The parsing results for Malt with predicted supertags would improve significantly as shown in Table 4.17 in comparison to Table 4.11. However it would be strange and expensive (time and memory-wise) to use the full PET parser as a supertagger, therefore this setup is not particularly realistic.

The results in Tables 4.11 and 4.17 indicate that DT is very sensitive to supertagging accuracy. For each PTB tag we collected corresponding supertags from the gold-standard training set. For open word classes such as nouns, adjectives, adverbs and verbs the relation between

PTB tags and supertags is many-to-many. Unique one-to-many correspondence holds only for possessive *wh*-pronoun and punctuation.

Thus, supertags do not provide just an extra level of detalization for PTB tags, but PTB tags and supertags are in part complementary. As discussed in Section 4.2, they contain bits of information that are different. For this reason their combination results in slight, but not always significant, increases of accuracy for all three parsers on all dependency formats (Tables 4.11 and 4.12, *PTB tags + supertags*, and Table 4.13, *Predicted PTB + gold supertags* and *Predicted supertags + gold PTB*). B&N predicts supertags with an accuracy of 92% for SB which is significantly lower than the state-of-the-art result of 95% (Ytrestøl, 2011).

When we consider *punctuation* in the evaluation, all scores improve, and in most cases significantly, for all parsers on DT. This is explained by the fact that punctuation in DT is always attached to the nearest token which is easy to learn for a statistical parser (Tables 4.14 and 4.16). Similar observations are made in the later work of Ma et al. (2014) who propose to treat punctuation marks as properties of context words for dependency parsing. They show that dependency parsers, trained on PTB-inspired analysis of punctuation (like SB, CD and PA in our experiments), achieve low accuracies on punctuation because heads of punctuation marks are not well-defined linguistically; however, removing all punctuation from parsing results in a loss of information important for dependency parsing.

We can also evaluate the effects of the *treebank size*. When less DeepBank data was available, we performed these experiments on the first 16 sections with a training set of 22,209 sentences and a test set of 1759 sentences (Ivanova et al., 2013b). The accuracy of the parsers in all configurations is higher when we train on more DeepBank data. However, for many of the current setups the difference with the previous results is not significant.

From the aspect of dependency scheme, the treebank size has the following impact:

- On DT all parsers show a significant improvement of accuracies in most of the setups on a larger treebank.
- On SB Malt and MST do not show significant changes in accuracy in most configurations with respect to the treebank size, whilst B&N has a significant rise of accuracy in the setups where it has to predict supertags when more data is available.
- On CD Malt, MST and B&N (in the configuration in which it predicts PTB tags) do not show significant changes of accuracy with respect to the treebank size.
- On the smaller amounts of data B&N performed better on SB and CD than on DT in the setup where it had to predict PTB tags. Availability of more data eliminate these differences in the B&N parser performance.

For the parsers we furthermore note an interesting effect of MST having a significant increase of accuracy in more setups than Malt on the new data.

For the PoS tags we can make the following observations:

- with gold tags (PTB and supertags) the MST and B&N parsers have significant performance improvements on DT as more data becomes available;
- Malt has significant improvements only on DT with gold PTB tags.

PoS tag	Description	SB	CD	DT	PA
CC	Coordinating conjunction	80%	83%	67%	54%
IN	Preposition or subordinating conjunction	83%	70%	76%	78%
\$	\$	90%	94%	93%	43%
CD	Cardinal number	94%	93%	91%	59%
NN	Noun, singular or mass	92%	93%	90%	84%
NNP	Proper noun, singular	95%	94%	92%	81%
NNS	Noun, plural	91%	92%	90%	82%
MD	Modal	98%	88%	83%	74%
VB	Verb, base form	83%	97%	97%	85%
VCN	Verb, past participle	78%	89%	90%	75%
VBG	Verb, gerund or present participle	74%	83%	82%	72%
VBP	Verb, non-3rd person singular present	86%	86%	84%	82%
VBZ	Verb, 3rd person singular present	88%	91%	85%	83%
JJ	Adjective	93%	95%	89%	92%
RB	Adverb	81%	70%	79%	81%

Table 4.18: Distribution of accuracy over PTB PoS tags when parsing with the Malt parser on the four dependency formats

- B&N benefits from being supplied new data on all three schemes when it predicts supertags.

Scoring punctuation had the following impact:

- for DT the scores improved significantly when punctuation was added while for SB and CD the scores decreased for all three parsers.

4.4 Error analysis

In order to better understand our results we perform an error analysis where we examine LAS for individual PoS tags and further analyze observed difficulties in parsing coordinations and verbs. We analyze the results of parsing with gold PoS tags. In addition we attempt to investigate which properties of the PA scheme make it particularly hard for parsers to process.

Table 4.18 shows the accuracy of the Malt parser on the frequent PTB PoS tags, however the tendency holds for other parsers as well. The first group are PoS from notoriously difficult linguistic structures: coordination and prepositional phrases; the second group are specific for WSJ: the dollar sign (\$) and cardinal numbers; the third group consists of nouns; the fourth group consists of verbs; the fifth group are adjectives; the sixth group are adverbs. We observe that the coordinating conjunction (CC) is particularly hard to parse with PA and DT analyses. As we recall from Chapter 3, coordination receives different treatments in our formats, see Figure 4.1. Coordinating conjunction is chosen as the head of the coordination structure in DT and PA, which has been shown by Schwartz et al. (2012) to be harder to parse with statistical parsers than analyses that select the first conjunct as the head. Table 4.18 shows the accuracy of the

Format	Precision	Recall
SB	0%	0%
CD	86.21%	89.26%
DT	74.77%	69.82%
PA	55.49%	42.71%

Table 4.19: Precision and recall of the labeling and attachment of the outgoing arcs for the coordinating conjunction (the “CC” PoS tag) for the Malt parser on different dependency formats with PTB PoS tags

incoming arcs for the coordinating conjunction with the PoS tag “CC”. Accuracy of detection and labeling outgoing dependency arcs is also different for the formats as demonstrated with Table 4.19. SB has on average zero outgoing arcs from coordinating conjunction, CD one, DT 2 and PA 3-4 which makes it harder to predict outgoing arcs for PA in contrast to DT and for DT in contrast to CD. SB has 0% precision and recall because by the structure of the coordination illustrated in Figure 4.1 a coordinating conjunction can never have outgoing arcs, and its confusion matrix on the test set is the following: the number of true positives is 0, the number of false negatives is 9 and the number of false positives is 12.

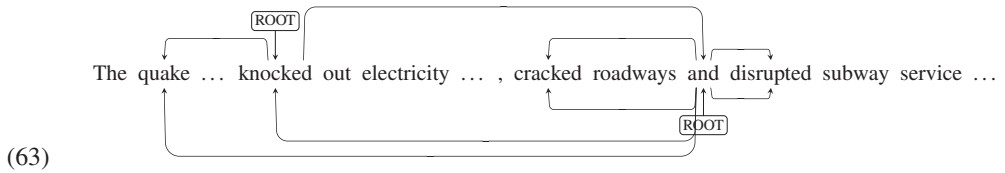
Though the approach used in PA and DT is harder for a parser to learn, it has some linguistic advantages: using SB and CD annotations, we cannot distinguish the two cases illustrated with the sentences (a) and (b) from WSJ:

- a) The fight is putting a tight squeeze on profits of many, threatening to drive the smallest ones out of business and straining relations between *the national fast-food chains and their franchisees*.
- b) Proceeds from the sale will be used for remodeling and refurbishing projects, as well as for *the planned MGM Grand hotel/casino and theme park*.

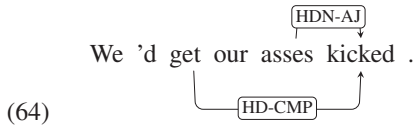
In the sentence a) “the national fast-food” refers only to the conjunct “chains”, while in the sentence b) “the planned” refers to both conjuncts and “MGM Grand” refers only to the first conjunct.

The Jaccard index values in Chapter 3 suggested that DT and CD are relatively similar to each other. Although overall labeled and unlabeled attachment scores did not show particular similarities in parsing DT and CD, the accuracies in Table 4.18 show that the performance of the parser on DT is closer to the performance on CD than SB and PA on most types of verbal PoS.

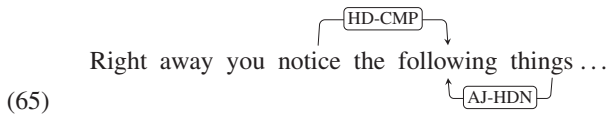
All parsers have high error rates on verbal PoS with all annotation formats. We analyzed the errors manually comparing analyses of Malt parser with DT to the gold standard analyses. The errors that concern attachment and labeling of verbs most commonly occur in longer sentences with multiple verbs and coordinating conjunctions and these errors are sometimes correlated with incorrect root detection (see Example (63) from WSJ where the top analysis is erroneous and the bottom analysis is correct).



In some cases the parser fails to detect longer dependencies: for the sentence in Example (64) from WSJ the parser suggests that the past participle “kicked” modifies the noun while in the gold standard it is a complement of the main verb.



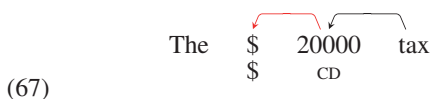
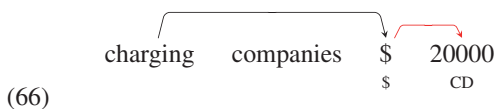
Many errors on gerund verb forms occur when the gerund is modifying a noun in the gold standard, see Example (65) from WSJ.



Our results from the previous section naturally raise the question of why the performance of the parsers is much lower with the PA format than with other dependency schemes. The analytical representation was originally developed for manual annotation of Czech, and later on exploited for Arabic (Hajič et al., 2004), languages from different language families than English. On the other hand, SB, CD and DT were originally designed for English.

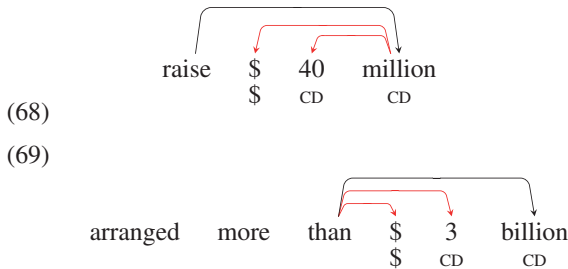
From Table 4.18 we can conclude that conversion from the phrase structures of the Penn Treebank into PA dependencies is noisy for some common WSJ constructions that include dollar sign and numbers like “\$ 3 billion”. While SB, CD and DT are rather consistent in choosing the dollar sign as the head of the number in most of these cases, the annotation of such structures in PA can vary depending on a context:

- a number can depend on a dollar sign, see Example (66);
- a dollar sign can depend on a number, see Example (67);
- both a dollar sign and a number can depend on the noun, see Example (68);
- both can depend on something else, see Example (69)



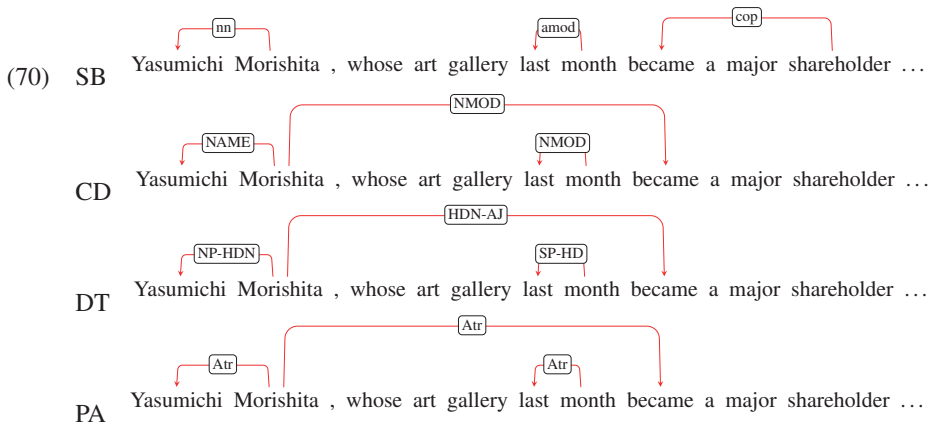
Format	Recall	Precision
SB	89.98%	89.98%
CD	93.52%	94.06%
DT	89.63%	89.63%
PA	83.84%	84.45%

Table 4.20: Recall and precision of dependency relation and attachment for root when parsing with the Malt parser on the four dependency formats with PTB PoS tags



PA furthermore takes a complex approach to the annotation of punctuation. The final punctuation is usually attached to root, as the precision and recall of dependency relation and attachment for root are significantly lower for PA than for other formats (see Table 4.20 for Malt parser; the tendency is similar for other parsers and other PoS tags), it introduces additional challenges for correct head detection for punctuation.

Labeling is another point where PA differs notably from the other formats in the fact that it has 17 different labels for the test set in contrast to SB, CD and DT that offer 49, 44 and 43 different labels correspondingly. Some labels in PA are in a sense overloaded: for example, the *Atr* label can be used for several different types of modifiers: a modifier in a complex noun phrase (“Yasumichi Morishita”), a head word in a modifying phrase (“became”) and an adjective in a noun phrase (“last month”), in contrast to other formats, see Example (70).



The PA format allows non-projective dependencies, but it does not seem to be a factor that complicates parsing since only 0.17% of the sentences in the training set and 0% of the sentences in the test set are analyzed as non-projective with PA. SB, CD and DT provide 0% of non-projective analyses on the training and test sets. The quantity of the long dependencies that the format introduces also does not seem to play a significant role in our case: PA does not have many more long dependencies than SB, CD and DT neither in general, nor on the PoS tags for which it has the worst accuracy.

To conclude, the parsers make very similar errors on SB, CD and DT. The most common problems concern coordination structures, preposition attachment and some types of verbs. LAS for individual PoS are in general closer for the pair CD and DT rather than for the pair SB and DT which can be an indirect indication of similarity of DT and CD in terms of parsing. The PA format appears to be harder to learn than SB, CD and DT which could be related to the fact that it was originally developed for morphologically rich languages and later adapted for English. Although parsers have more difficulty to learn the analysis of coordination, attachment of prepositions and other phenomena on PA than on other formats, the major issue is related to PA permitting variations of the analysis of the dollar sign + number constructions which are exceedingly common in the WSJ corpus.

4.5 Extrinsic evaluation

Syntactic analysis is usually an intermediate step in a larger natural language processing task. Using a downstream application, we are interested to explore whether the small parsing differences between CD, SB and DT affect the overall performance of the system and additionally test our claim that DT constitutes a reasonable choice of dependency representation compared to CD and SB. Elming et al. (2013) showed that the choice of dependency format can have considerable impact on end results of downstream applications, including SMT and negation resolution.

We perform an extrinsic evaluation using negation resolution (NR) as our downstream evaluation task. The negation resolution system that we use conditions its decisions on both dependency arcs and labels and it was sensitive to the choice of the annotation scheme in the experiments of Elming et al. (2013).

NR is the task of determining, for a given sentence, which tokens are affected by a negation cue (Lapponi et al., 2012a). A negation *cue* is an operator of negation, i.e. a word or a combination of words that expresses negation (for example, “no”, “not”, “never”, “no longer”, “by no means”). The *scope* of negation is *the part of a sentence that is negated* (Huddleston and Pullum, 2002) and it includes all negated concepts. The *negated event* is the event/state inside the scope which is semantically negated (Lapponi et al., 2012a). In the specific pre-existing task data for negation scope resolution that we use for our extrinsic evaluation, only factual events can be negated: events are absent in instances of imperatives, conditionals, suppositions etc. Example (71) shows the negation cue marked in bold letters, the scope enclosed in square brackets and the negated event underlined.

(71) [John had] **never** said as much before].

	Train	Dev	Test
# tokens	65,450	13,566	19,216
# sentences	3644	787	1089
# sentences with negation	848	144	235
# cues	984	173	264
# scopes	887	168	249
# negated events	616	122	173

Table 4.21: Conan Doyle corpus statistics (Morante and Blanco, 2012)

Software and Data

For our experiment we use the negation resolution system UiO₂ (Lapponi et al., 2012b) that was one of the best performing systems of the 2012 Computational Semantics (*SEM) shared task (Morante and Blanco, 2012). UiO₂ is a CRF-based sequence-labeling system for negation scope resolution relying on syntactic information from dependency graphs. The dataset is composed of negation annotated stories of Conan Doyle prepared for the 2012*SEM shared task (see corpus statistics in Table 4.21).

The gold cues are being used in this experiment on the development set and predicted cues are used on the test set. This means that if the sentence in Example (71) occurred in the development set, the system would have to identify “*John had*” and “*said as much before*” as scopes and “*said*” as the negated event provided that “*never*” is already known to be a cue. However if the sentence in Example (71) occurred in the test set, the system would first have to identify “*never*” as a cue.

We use the evaluation software developed by the 2012*SEM shared task organizers (Morante and Blanco, 2012) which applies the following metrics, ignoring the punctuation tokens:

- Cue-level F_1 -measure (*Cues*)
- Scope-level F_1 -measure (*Scopes*)
- F_1 -measure over scope tokens (*Scope tokens*). The total of scope tokens in a sentence is the sum of tokens of all scopes. For example, if a sentence has two scopes, one of five tokens and another of seven tokens, then the total of scope tokens is twelve
- F_1 -measure over negated events (*Negated*), computed independently from cues and from scopes
- Global F_1 -measure of negation (*Global*): the three elements of the negation - cue, scope and negated event - all have to be correctly identified (strict match)
- Percentage of correct negation sentences (*CNS*). This is the strictest evaluation measure reported by the evaluation script, as there can be multiple instances of negation in one sentence.

	B&N-CD	B&N-SB	B&N-DT	Malt-SB	Mate-yamada	Mate-conll07	Mate-ewt	Mate-lth
Scopes	79.57	81.69	80.43	80.43	81.27	80.43	78.70	79.57
Scope tokens	87.47	87.77	88.13	86.42	-	-	-	-
Negated	75.96	71.15	73.33	79.44	76.19	72.90	73.15	76.24
Global	65.89	63.78	65.89	66.92	67.94	63.24	61.60	64.31
CNS	45.83	45.83	47.92	47.92	-	-	-	-

Table 4.22: Performance of the negation resolution system UiO₂ (Lapponi et al., 2012b) on the development set of the Conan Doyle corpus from the 2012*SEM shared task with gold cues using dependency features from the analyses of B&N with CD, SB and DT dependencies in contrast with previous work: (a) the analyses of the Malt parser with SB (Lapponi et al., 2012b) and (b) the analyses of the Mate parser with yamada, conll-07, ewt and lth dependencies (Elming et al., 2013)

Results and discussion

In order to obtain syntactic features for the negation resolution system, we employ B&N with CD, SB and DT representations. The choice of the parser is motivated by the experiments in Section 4.3 in which this parser shows the highest dependency accuracy scores. We run the negation resolution system with gold cues.

Table 4.22 presents the results of running the NR system on the analyses produced by B&N with three annotation schemes, e.g. CD, SB and DT and also includes results from previous work with the Malt parser on Stanford Basic dependencies (Lapponi et al., 2012b) and with the Mate parser on yamada, conll07, ewt and lth dependencies (Elming et al., 2013)². The previous work results are not directly comparable to our experiments with B&N because the models in Lapponi et al. (2012b) are trained on sections 2-21 of the WSJ extended with about 4000 questions from the QuestionBank (Judge et al., 2006), the models in Elming et al. (2013) are trained on sections 2-21 of WSJ while the models of B&N in our work are trained on sections 0-19 of the WSJ.

As observed in Elming et al. (2013), we get differences of performance across the three formats: with B&N, CD appears to perform best for detection of negated events, SB for scopes while DT shows the best performance on scope tokens, global negation and correct negated sentences. Compared to the scores in the previous work, with B&N on DT we obtain better performance on scopes and scope tokens and lower results for negation events, global negation and correct negation sentences. Taking into account that the training set for the previous work of Lapponi et al. (2012b) was larger, our results seem promising.

The extrinsic evaluation results support our hypothesis that the DT format is relevant for parsing and applications relying on syntactic information from bilocal dependencies on par with the commonly used CD and SB formats.

²CD, conll07 and lth annotation schemes are obtained from PTB by running the LTH constituent-to-dependency conversion tool of Johansson and Nugues (2007) with different command-line options.

4.6 Summary

In this chapter we performed an empirical comparison of four dependency schemes, three statistical parsers and two part-of-speech tag sets. We investigated which properties of the annotation formats are harder to learn with statistical parsers, how accurate the parsers are with respect to each other in terms of accuracy and how the choice of the lexical types affects the performance of the syntactic analyzers.

The main conclusions of the chapter are that SB, CD and DT are nearly equally hard to learn with B&N when punctuation is not scored (i.e. the difference in the accuracy for DT and at least one of these two schemes is statistically insignificant in almost all of the setups), and DT is the easiest format for this parser when punctuation is considered in the overall accuracy values. B&N is more accurate than Malt and MST in our setups while the Malt parser is the fastest among the three. The traditional PTB PoS tags naturally fit the SB and CD schemes well, while the larger set of ERG lexical types is unsurprisingly more suitable for the DT format. The PA annotations appear to be more difficult due to a very frequent construction in WSJ text that includes the dollar sign and a number.

As Miwa et al. (2010) showed that intrinsic and extrinsic evaluations do not always correlate, it is interesting to contrast DT with the other schemes on a downstream application. We have chosen negation resolution, the task of identifying tokens affected by a negation cue, because this NLP task is sensitive to the differences in the input dependency annotation as shown in Elming et al. (2013). Using the CRF-based system of Lapponi et al. (2012b) we compare DT to SB and CD and conclude that each of them optimizes different evaluation metrics with no single winner, though DT is best such metrics as scope tokens, global negation and correct negated sentence.

Chapter 5

Cross-framework parser evaluation

In this chapter we compare three approaches to automatic syntactic analysis for English: data-driven dependency parsing, statistical phrase structure parsing and a hybrid, grammar-based parsing. These three parsing architectures differ in the underlying theories introduced in Chapter 2 and produce different structures. We use syntactic bilexical dependencies as a common basis for comparison. Dependency-based evaluation (Lin, 1998; Carroll et al., 1998) is a common cross-platform evaluation approach where the heads, that play a central role in dependency grammar, have a key part and can be recovered in computational frameworks based on HPSG or some version of X' theories. In our study we observe which trade-offs apply along three dimensions—accuracy, efficiency, and resilience to domain variation. We show that the hand-built grammar helps in accuracy and cross-domain syntactic parsing performance, but that there are no significant differences in results for data-driven and grammar-based parsers in extrinsic evaluation on the negation resolution task.¹

5.1 Motivation

In the present work we seek to perform head-to-head comparison of the ERG parser to the state-of-the-art data-driven syntactic analyzers. Grammar-based systems for natural language processing rely on detailed linguistic knowledge incorporated in formal grammar with constraints. The underlying linguistic grammar theory is expressed in the form of rules that describe the valid composition of linguistic entities. The result consists of all possible syntactic analyses of the sentence. Unlike grammar-based systems, data-driven systems do not perform exhaustive linguistic analysis but rather rely on statistics and generalizations and are conventionally assumed to be more robust and efficient than grammar-based systems.

A contrastive study requires data annotated in native parsers' formats and a framework-independent evaluation method. For phrase structure and dependency parsers the WSJ portion of the Penn Treebank (Marcus et al., 1993) has been the predominant resource for English. Availability of the ERG annotation layer over the WSJ text (Flickinger et al., 2012) makes it possible to include the HPSG PET parser into the comparison. For the type of HPSG analyses recorded in DeepBank, we applied a reduction to projective bilexical dependencies, Derivation Tree-Derived Dependencies (DT) (Zhang and Wang, 2009; Ivanova et al., 2012) that were

¹This chapter is an extended version of Ivanova et al. (2013a) and Ivanova et al. (2015).

discussed in detail in Section 3.4.1.

Dependencies are recognized as an attractive target representation for syntactic analysis and have been repeatedly used for cross-formalism parser evaluation. We choose an experimental setup like the one of Cer et al. (2010): first, parsing into the representation native for the given parser and then deterministically reducing the structures to bilexical dependencies. In earlier work, e.g. of Clark and Curran (2007a) and Miyao et al. (2007), the comparison of different parsing approaches required a mapping between the annotation innate for a parser and the target representation used as base for comparison. We argue that such conversions inevitably introduce fuzziness into the comparison since they are based on heuristic rules and require changes of lexical heads.

Large-scale grammars have been designed to deliver precise linguistic analysis and provide broad coverage. We are interested in how the accuracy and coverage of a grammar-based parser correlate with the state-of-the-art statistical analyzers. Time and space efficiency have long constituted problems for grammar-driven parsing technologies though technological and scientific breakthroughs have enabled overall run-time efficiency gains of a factor of around 2000 and a reduction of the space consumption by several orders of magnitude². Nevertheless grammar parsing is still considered to be slower than contemporary statistical processing; in this chapter we are going to evaluate how large this gap is. In addition, we seek to document how the trade-offs in terms of parser accuracy and efficiency are affected by domain and genre variation, which is possible due to the availability of the corpora instantiating a comparatively diverse range of domains and genres annotated with ERG analyses (Oepen et al., 2004) in combination with the conversion procedure described in Chapter 3.

5.2 Related work

Cross-framework parser comparison has continually attracted research interest and was in focus of dedicated workshops such as the Workshop on Cross-Framework and Cross-Domain Parser Evaluation (Bos et al., 2008). As we have already discussed in Section 2.4, adopting dependency-based evaluation methods (Lin, 1998) is a commonly used approach to framework-independent parser comparison.

Among dependency-based evaluation procedures, grammatical relations (GR) (Carroll et al., 1998) have been the target of a number of benchmarks employing the PARC 700 DepBank (King et al., 2003), a corpus of predicate-argument relations for 700 sentences from section 23 of the WSJ constructed for evaluation purposes. The resource was created semi-automatically by the automatic conversion of deep parser output and manual correction of the structures. The parsing system used to build the analyses is the XLE system (Maxwell and Kaplan, 1993) with the underlying LFG formalism (Kaplan and Bresnan, 1982); only f-structures are used in the conversion. A disadvantage of the GR-based method is that it requires a heuristic mapping from “native” parser output to the target representation for evaluation, which makes the results hard to interpret. The transformations are often non-trivial and require modifications in the choice of heads and other adjustments (Kaplan et al., 2004; Clark and Curran, 2007a; Miyao et al., 2007). The upper bounds of parser performance on GR are 84.76% for CCG (Clark and

²<http://www.delph-in.net/wiki/index.php/Background> Accessed: 14 August 2015.

Curran, 2007a), 84.27% for Enju and 73.21% for C&J, Charniak and Stanford parsers (Miyao et al., 2007) (upper bounds are computed by comparison of gold standard treebanks converted to GR against DepBank). The nearly 20% loss of accuracy due to format transformations can swamp any actual parser differences (Miyao et al., 2007).

Another dependency representation commonly applied for formalism-independent parser comparison is the Stanford Dependency scheme (de Marneffe et al., 2006), which is inspired by GR and is similar in concept. Although there is no hand-annotated treebank available, a program for conversion of the PTB trees into SD relations is incorporated into the Stanford parser. The advantages of SD is that the conversion from CFG trees is deterministic and comparisons of phrase structure and direct dependency parsers can be run on PTB data and therefore related to the previous work on parsing. However, for parsers producing formats other than phrase structure and dependency, similar challenging and fuzzy heuristic mappings as for GR are required (Miyao et al., 2007). Unlike SD that was proposed outside the deep parsing community and may lack the potential to assess deep grammar parsers, the dependency formalism that we use, DT is defined in terms of a formal linguistic theory and directly captures decisions taken in the grammar.

At the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, PropBank was mentioned as a candidate gold standard for cross-framework parser comparison (Bos et al., 2008), however, organizers noted that it is not commonly used for parser evaluation. The advantage of PropBank is that it is developed independently of any specific grammar formalism. Gildea and Jurafsky (2002) and Gildea and Palmer (2002) built a system to predict FrameNet and PropBank semantic roles correspondingly from phrase structure trees produced by the statistical Collins parser (Collins, 1997). Their work shows that improvements in parsing translate into better semantic interpretations. Burke et al. (2005) evaluate automatically predicted f-structures of LFG against PropBank. Their method employs an automatic conversion of PropBank annotations to bilexical dependencies and a mapping of f-structures to the PropBank-style semantic annotations in dependency format. The reported upper bound of precision for the PropBank, computed using gold trees, is 71.1%. The authors described some challenges of the mapping of f-structures to PropBank triples and pointed out the demand of a universal set of gold standard PropBank triples. Among the probable reasons why PropBank is not commonly employed in parser comparison are that annotations are outside the limits of the grammatical level to facilitate parser evaluation (Bos et al., 2008) and that its annotations concern only semantic roles of verbs and thus offer a fragmental dependency analysis for a sentence as we have seen in Figure 3.7a in Chapter 3.

There has been a number of studies on parser comparison in two main directions: (a) contrasting phrase structure to dependency parsing and (b) contrasting hybrid grammar-based approaches to purely statistical parsing methods. Cer et al. (2010) examined the performance of several dependency parsers versus phrase structure parsers in terms of speed and accuracy using the SD scheme. They found that phrase structure parsers with output post-conversion to dependencies are more accurate than direct dependency parsers, while some of the latter ones are 40%-60% faster. However most phrase structure parsers allow users to trade off accuracy for speed, and a best balanced configuration that is both fast and more accurate than direct dependency parsers is achievable for the constituent Charniak-Johnson parser. The effects of domain shifts are not addressed in this work.

Fowler and Penn (2010) formally showed that a wide range of CCGs are context-free. They then trained the PCFG Berkeley parser on CCGBank (Hockenmaier and Steedman, 2007) and achieved better scores in terms of supertagging accuracy, PARSEVAL measures and dependency accuracy than with the CCG parser. They thus conclude that the specialized CCG parser is not necessarily more accurate for parsing with CCG than the general-purpose Berkeley parser. Since the derivation tree of the ERG can be represented in the form of constituency tree, we contrast the performance of the Berkeley parser and the deep ERG parser in a similar manner. The study of Fowler and Penn (2010), however, fails to also take parser efficiency into account.

In a related work for Dutch, Plank and van Noord (2010) suggest that, intuitively, one should expect grammar-driven systems to be more robust to domain shifts than purely data-driven systems. They substantiated this expectation by showing that the HPSG-based Alpino system performs better and is more resistant to domain variation on parsing into Dutch syntactic dependencies than the direct dependency parsers Malt and MST.

It has been repeatedly shown in previous work that depending on the choice of downstream application, the results of intrinsic and extrinsic evaluation might not correlate. Intrinsic evaluation provides assessment of a parser as an independent unit whereas extrinsic evaluation demonstrates the effects of the choice of a particular parsing system as a module of some application. These two analyses of the quality of a parser are not always in line.

For example, Mollá and Hutchinson (2003) argue about the limited value of intrinsic evaluation for application benchmarking. They find that the difference in answer extraction performance with two different dependency parsers is less drastic and does not agree on some metrics with the results of intrinsic evaluation based on grammatical relations.

Miyao et al. (2008) were first to show that the accuracy of an application is similar for different WSJ-trained parsers drawn from a variety of distinct frameworks: phrase structure, dependency and grammar-based parsing. The authors experiment with several parsers and dependency formats in application to the protein-protein interaction (PPI) extraction from biomedical text. However, it is worth noting that parsing speed is significantly different which affects the efficiency of the whole system. While in the experiments of Miyao et al. (2008) the dependency parsers are the fastest, followed by the grammar parsers and the phrase structure parsers are relatively slower, in the present work a slower dependency parser is employed which results in different ranking with respect to speed.

5.3 Experimental setup

This section presents resources and tools employed in our study and describes how we tuned statistical parsers.

Resources and tools

Data The in-domain experiments are carried out on the DeepBank data with the splits suggested by the developers of this resource: sections 0-19 for training, section 20 for tuning and section 21 for testing (WSJ). As we recall from Section 2.5, DeepBank in its version 1.0 lacks analyses for some 15 percent of the WSJ sentences, for which either the ERG parser failed to

		Sentences	Tokens	Types
DeepBank	Train	33,783	661,451	56,582
	Tune	1,721	34,063	8,964
	WSJ	1,414	27, 515	7,668
Redwoods	CB	608	11,653	3,588
	SC	864	13,696	4,925
	VM	993	7,281	1,007
	WS	520	8,701	2,974

Table 5.1: Sentence, token, and type counts for the DeepBank and Redwoods data sets

	LAS	UAS
CD	90.92	93.97
SB	90.65	93.05
DT	90.91	93.18

Table 5.2: Parsing accuracy of B&N with three dependency schemes, measured as labeled attachment score (LAS) and unlabeled attachment score (UAS), excluding punctuation symbols

suggest a set of candidates (within certain bounds on time and memory usage), or the annotators found none of the available parses acceptable. For cross-domain tests the following parts of the Redwoods treebank were used: an early essay on open source software “Cathedral and the Bazaar” by Eric Raymond (CB); a fragment of the SemCor corpus which is a sense-tagged corpus with the Brown Corpus as the text and WordNet as the lexicon (SC); resources from the VerbMobil corpus, a collection of transcribed spontaneous speech recorded in a dialogue task (VM); a part of the Wikipedia-derived WeScience Corpus (WS). Table 6.1 provides the summary of sentence, token and type counts for the data sets.

It is fair to mention that the CB and VM datasets have been used in ERG development and for this reason the linguistic phenomena occurring in these texts are probably well-covered in the grammar, which might give some advantage to PET over statistical parsers. On the other hand, SC has hardly had a role in grammar engineering, and WS and section 21 of the Wall Street Journal (WSJ) are absolutely unseen and are entirely new for the PET parser trained on ERG version 1212, used in the experiments below.

Dependency format The parsers are compared on the projective DT dependency representation grounded in the formal HPSG theory of grammar as described in Chapter 3. The adoption of this format is justified by the observations from the two previous studies. Firstly, the structural analysis in Chapter 3 showed that this format corresponds to the basic variant of Stanford Dependencies and to the CoNLL scheme from the dependency parsing competitions with the strongest Jaccard similarity of unlabeled dependencies for the pair DT and CoNLL. Secondly, the empirical study in Chapter 4 that investigated accuracy levels available for the different target representations when training and testing a range of state-of-the-art parsers on the same data sets, demonstrated that differences in parsing DT and the commonly used Stanford Basic (SB) and CoNLL (CD) representations are generally small (see Table 5.2) .

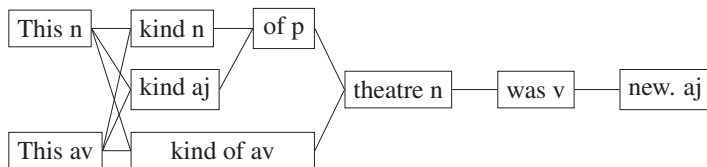


Figure 5.1: Ambiguous token lattice

Parsers The parsers that were trained and run in the series of experiments below are the grammar-based PET parser (Callmeier, 2000), the dependency B&N parser (Bohnet and Nivre, 2012) and the Berkeley phrase structure parser (Petrov et al., 2006).

There are two setups for PET: one optimized for accuracy and the other one optimized for efficiency. The variant with accuracy optimization, henceforth ERG_a , is trained with the default configuration used by the DeepBank and Redwoods developers. The more efficient variant, ERG_e , is the best-performing configuration of Dridan (2013a) with lattice-based sequence labeling over lexical types and lexical rules. In this setup of the parser supertagging is performed over ambiguous tokenization using a model similar to a standard trigram Hidden Markov Model. A simplified example of an ambiguous token lattice over which the model operates is shown in Figure 5.1 (Dridan, 2013b). Aforesaid integration of generalization of supertagging, so-called *ubertagging*, with PET offers speed-up at marginal loss in accuracy.

Unlike the other parsers in our experiments, PET does not have a trivial interface to accept pre-tokenized input. We feed the parser raw strings and it internally operates over an ambiguous token lattice. To emulate the effects of gold-standard tokenization, the parser is requested to generate a 2000-best list of sentence analyses which is then filtered for the top-ranked analysis matching the gold treebank tokenization. This approach is imperfect, as in some cases no token-compatible analysis may be on the n-best list, especially so in the ERG_e setup (where lexical items may have been pruned by the sequence-labeling model). When this happens, we fall back to the top-ranked analysis and adjust our evaluation metrics to robustly deal with tokenization mismatches.

B&N was chosen because it achieves higher levels of tagging and parsing accuracies than pipeline systems Malt and MST for typologically diverse languages such as Chinese, English, and German. Our experiments on parsing different dependency schemes in Chapter 4 showed that B&N is also less sensitive to the choice of dependency format than Malt and MST. We apply this parser mostly out-of-the-box with an increased beam size of 80.

As a context-free grammar parser we chose the Berkeley parser which is currently one of the best-performing (and relatively efficient) PCFG engines. Since Berkeley requires training on phrase structure trees, we used ERG derivations of HPSG analyses that can be looked at as context-free trees. For technical reasons, transformation from ERG to PTB tokenization is not applicable in the setup with context-free trees therefore we limited experiments involving Berkeley to ERG tokens. The Berkeley parser was run in an accurate mode in our experiments.

Raw										
i	n		t	h	e		'	7	0	s
0	1	2	3	4	5	6	7	8	9	10
Gold						Parsed				
in the '70s						in the '70s				
< 0, 2, in >						< 0, 2, in >				
< 3, 6, the >						< 3, 6, the >				
< 7, 8, ' >						< 7, 11, '70s >				
< 8, 11, 70s >										

Figure 5.2: Character ranges for the phrase “in the ’70” which is tokenized differently in parser output with respect to the gold standard

Evaluation We apply evaluation metrics standard for dependency parsing, e.g. UAS and LAS (see Section 2.4), which we complement with tagging accuracy scores (TA), where appropriate. However application of the standard CoNLL eval.pl scorer is complicated due to tokenization mismatches in PET outputs related to the fact that PET cannot be forced to use gold tokenization as discussed before. In order to use eval.pl we would have to consider sentences with any tokenization mismatch as parse failure but in this scenario PET will be over-penalized since some correct dependencies will be eliminated from the parser output. For this reason, we apply another syntactic evaluation tool that does not require sentence segmentation and tokenization to exactly match the gold standard - Robust Evaluation of Syntactic Analysis (RESA) (Dridan and Oepen, 2013). Internally, RESA uses character-based token indexing from the raw text. The phrase “in the ’70s” in Figure 5.2 is tokenized differently by the parser with respect to the gold standard. The tokens are represented by their start and end character positions in the raw text. The parser analyzes “’70s” as one entity which spans from positions 7 to 11 in the raw string, while in the gold standard “’70s” is split into tokens “’” and “70s” with the corresponding character ranges <7,8> and <8,11>. RESA carries out evaluation over triples of start character position, end character position and dependency label, and produces the same scores as eval.pl when there are no segmentation differences.

Tuning

B&N The choices about the level of detail conveyed by DT dependency labels and PoS tags made in Section 4.2 maximize the accuracy of the Malt parser. The goal of the tuning is to find out whether B&N can successfully use more information from the HPSG constructions or whether the fine distinctions result in sparseness and diminished accuracy. Table 5.3 demonstrates that complete ERG constructions do not give an advantage over short versions in terms of accuracy, therefore further experiments with B&N are carried out on the DT scheme consistent with descriptions in Chapters 3 and 4.

LAS	UAS	LACC	Coverage
long labels			
86.83	89.69	89.76	100
short labels			
87.29	90.02	90.07	100

Table 5.3: Tuning of B&N on section 20 of DeepBank. Evaluation performed with eval.pl. “Long labels” means that B&N is trained with the full ERG constructions; “short labels” means that B&N is trained with the ERG constructions cut on the first underscore

Berkeley Due to its ability to internally rewrite node labels, the Berkeley parser should be expected to adapt well to ERG derivations that compactly store information required to reconstruct the full HPSG analysis in the form of the phrase structure trees. Compared to the phrase structure annotations in the PTB, there are two structural differences. First, the inventories of phrasal and lexical categories are larger, at around 250 and 1000, respectively, compared to only about two dozen phrasal categories and 45 parts of speech in the PTB³. Second, ERG derivations contain more unary (non-branching) rules, recording for example morphological variation or syntactico-semantic category changes⁴.

Table 5.4 shows experiments to tune the Berkeley parser by choosing whether to preserve or skip unary productions, and whether to implement original ERG productions (“long”) for phrase categories or their generalizations as used in DT (“short”) or to keep long labels for unary rules and a variant with short labels for branching rules (“mixed”). We report results for training with five and six split–merge cycles, where fewer iterations led to less parse failures (“gaps” in Table 5.4).

Preserving unary rules seems favorable for the Berkeley parser, while for the length of the phrase categories there are apparent trade-offs: by choosing long labels over short ones we give up some coverage (“gaps” in Table 5.4) but gain in labeled accuracy. We do not observe a single best-performer, but as our primary interest across parsers is dependency accuracy, we select the configuration with unary rules and long labels, trained with five split–merge cycles, which seems to afford near-premium LAS at near-perfect coverage. In addition, this setup does not require modifications in the native ERG derivations.

³Examples of phrasal categories: PTB - ADJP (adjective phrase), NP (noun phrase), PP (prepositional phrase; examples); ERG - sb-hd_mc_c (subject-head, main clause), sb-hd_nmc_c (subject-head, embedded clause). Examples of lexical categories: PTB - NNS (noun, plural), JJR (adjective, comparative); ERG - n_pl_olr (plural noun with “s” suffix), n-nh_j-cpd_c (compound from noun and adjective).

⁴Examples of unary rules: PTB - S (unary in a sentence where the noun phrase was left unspecified in the original clause), NP (unary recursive rule); ERG - v_pas_odlr (morphological rule, past-participle), v_n3s-bse_ill (morphological rule, non-third person singular or base inflection), hdn_bnp-pn_c (bare noun phrase with a proper noun).

Labels Cycles		Unary Rules Removed			
		Long		Short	
		5	6	5	6
Gaps	3	3	0	0	
TA	88.46	87.65	89.16	88.46	
F ₁	74.53	73.72	75.15	73.56	
LAS	83.96	83.20	80.49	79.56	
UAS	87.12	86.54	87.95	87.15	

Labels Cycles		Unary Rules Preserved					
		Long		Short		Mixed	
		5	6	5	6	5	6
Gaps	2	5	0	0	11	19	
TA	90.96	90.62	91.11	91.62	90.93	90.94	
F ₁	76.39	75.66	79.81	80.33	76.70	76.74	
LAS	86.26	85.90	82.50	83.15	86.72	86.16	
UAS	89.34	88.92	89.80	90.34	89.42	88.84	

Table 5.4: Tagging accuracy (TA), PARSEVAL F₁, and dependency accuracy (LAS and UAS) for the Berkeley parser on the development data. “Long” means that Berkeley is trained with the full ERG constructions; “short” means that Berkeley is trained with the ERG constructions cut on the first underscore; “mixed” means that Berkeley is trained with the the full ERG constructions for unary rules and the ERG constructions cut on the first underscore for branching rules; “gaps” stands for the number of sentences for which Berkeley does not produce an analysis; 5 and 6 stand for the number of split-merge iterations

	Gaps	Time	TA _c	LAS _c	UAS _c
Berkeley	1+0	1.0	92.9	86.65	89.86
B&N	0+0	1.7	92.9	86.76	89.65
ERG _a	0+0	10	97.8	92.87	93.95
ERG _e	13+44	1.8	96.4	91.60	92.72

Table 5.5: In-domain parsing experiments. Parse failures and token mismatches (“gaps”), efficiency (“Time”) measured in seconds per sentence, tagging accuracy (“TA_c”) and dependency accuracy (“LAS_c” and “UAS_c”) on WSJ. Index _c means that evaluation is performed with the RESA software using character-based token indexing from the raw text

5.4 In-domain parsing results

Using the experimental setup described above, we perform a set of experiments where we compare Berkeley, B&N and the two setups for PET, ERG_a and ERG_e, in terms of coverage, efficiency and accuracy on WSJ. Table 5.5 provides a summary of the in-domain (section 21 of WSJ) cross-paradigm comparison of the three parsers. The table shows trade-offs in coverage, time, tagging accuracy and labeled and unlabeled dependency accuracies.

There are two sources of incomplete coverage: parse failures and differences in tokenization in parse output with respect to the gold standard. The second problem is peculiar only to the ERG parser, since it operates on raw strings while B&N and Berkeley receive gold tokens as

input. B&N and ERG_a deliver complete coverage, Berkeley experiences one parse failure, whereas ERG_e cannot parse 13 sentences (close to one percent of the test set), and we observe 44 inputs where parser output deviates in tokenization from the treebank. This is likely an effect of the lexical pruning applied in this setup. Berkeley is the fastest parser in our setup and ERG_a is one order of magnitude slower. The joint segmentation and supertagging setup of Dridan (2013a) leads to a speed-up of almost a factor of six and helps to achieve parsing speed comparable with B&N. In terms of tagging and dependency accuracies, the two statistical data-driven parsers perform very similarly. With the two setups of the grammar-driven parser there is about 1.3% accuracy loss along all the three metrics for the faster ERG_e due to the supertagging constraints on the parser search space. However, the most important result is that both setups of the ERG parser have significantly higher accuracies than B&N and Berkeley, with ERG_a resulting in 6% increase in labeled accuracy score.

5.5 Cross-domain parsing results

This section presents cross-domain experiments on the datasets from the Redwoods treebank that aim to test the domain-resilience of the parsers in question and ascertain the ERG parser gains in dependency accuracy over data-driven parsers. Table 5.6 introduces coverage, tagging accuracy, labeled and unlabeled attachment scores for the four parsers applied to the four new data sets without domain adaptation.

The first and most obvious observation is that for all the parsers domain shift adversely affects accuracy scores, an expected and well-known effect (Gildea, 2001). However, the important fact is that the drop is significantly smaller for the ERG parsers in comparison to the data-driven parsers across all the domains. As in the in-domain experiments, the Berkeley and B&N parsers perform similarly to each other. At the same time accuracies of ERG_a and ERG_e are relatively close to each other and notably higher than for the data-driven parsers. There is a rather clear distinction between “easier” and “harder” domains for Berkeley and B&N: both have the smallest labeled dependency accuracy loss (less than 6.4 points) on the texts from the WeScience corpus (WS) and the largest gaps (more than 11.4 points) for the spontaneous speech transcriptions (VM). For ERG_a and ERG_e the differences in accuracy scores are less pronounced, though the ERG parsers also tend to perform better on WS and experience most tokenization mismatches on VM. The tagging accuracies also indicate that the ERG parsers are more resilient to domain variation than B&N and Berkeley, and that WS is the easiest domain among the four for all the parsers, and VM is, on the other hand, the hardest. While B&N achieves complete coverage, Berkeley experiences one parse failure on CB and SC just as on the in-domain data, and 7 parse failures on VM and WS. It is interesting to note that ERG_e has less parse failures in the cross-domain experiments than in the in-domain test. The possible reason for that can be that the average sentence length is shorter for the out-of-domain data compared to WSJ.

From the out-of-domain results, it is evident that the general linguistic knowledge available in ERG parsing makes it far more resilient to variation in domain and text type. However, we did not perform domain adaptation for the statistical parsers.

		Gaps	TA _c	LAS _c	UAS _c
CB	Berkeley	1+0	87.1	78.13	83.14
	B&N	0+0	87.06	77.70	82.96
	ERG _a	0+4	96.3	90.77	92.47
	ERG _e	8+8	95.3	90.02	91.58
SC	Berkeley	1+0	87.2	79.81	85.10
	B&N	0+0	85.9	78.08	83.21
	ERG _a	0+0	96.1	90.84	92.65
	ERG _e	11+7	94.9	89.49	91.26
VM	Berkeley	7+0	84.0	74.40	83.38
	B&N	0+0	83.1	75.28	82.86
	ERG _a	0+40	94.3	90.44	92.27
	ERG _e	11+42	94.4	90.18	91.75
WS	Berkeley	7+0	87.7	80.31	85.11
	B&N	0+0	88.4	80.63	85.24
	ERG _a	0+0	97.5	91.33	92.48
	ERG _e	4+12	96.9	90.64	91.76

Table 5.6: Cross-domain parsing experiments. Coverage (parse failures and token mismatches) and tagging and dependency accuracies on Redwoods test data. Attachment scores are calculated including punctuation symbols

5.6 Error analysis

The ERG parsers outperform the two data-driven parsers on the WSJ data. Through in-depth error analysis, we seek to identify parser-specific properties that can explain the observed differences. In the following, we look at (a) the accuracy of individual dependency types, (b) dependency accuracy relative to (predicted and gold) dependency length, and (c) the distribution of LAS over different lexical categories for Berkeley, B&N and PET (ERG_a).

The histograms in Figures 5.3-5.7 present the distribution of the most frequent dependency labels and the accuracy of the frequent individual dependency types on the in- and out-of-domain datasets. We observe that over all domains, the relations *hd-cmp* (head-complement) and *sp-hd* (specifier-head) are among the 5 most frequent and accurate. The notion of an adjunct is difficult for all three parsers across domains as two of the most error-prone dependency labels are the *hd-aj* (post-adjunction to a head) and *hdn-aj* (post-adjunction to a nominal head) relations employed for relative clauses and prepositional phrases. The most common error for the latter relation is verbal attachment.

Dependency length is the distance between a head and its argument. It has been noted that dependency parsers may exhibit systematic performance differences with respect to dependency length (McDonald and Nivre, 2007). Figures 5.8-5.12 show dependency label precision and recall relative to dependency length.

Dependencies spanning many tokens have different frequencies on different domains: for example, dependencies with length 21-30 comprise 0.4% of WSJ, 0.26% of CB, 0.3% of SC 0% of VM and 0.29% of WS. Sentences on VM are very short and this dataset contains twice as

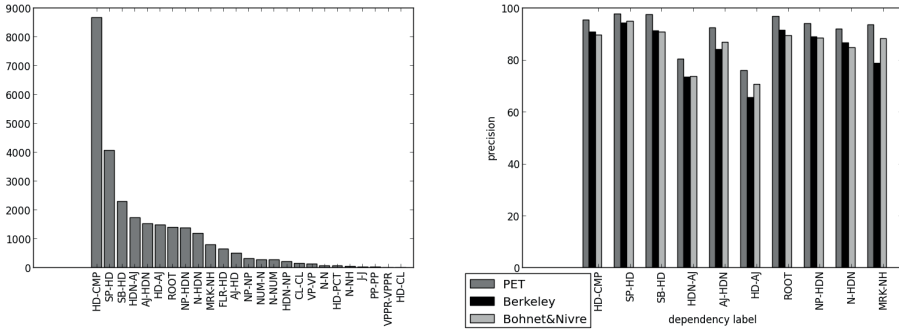


Figure 5.3: Domain: WSJ. (Left) Distribution of dependency labels; (right) precision of attachment of the most frequent dependency types

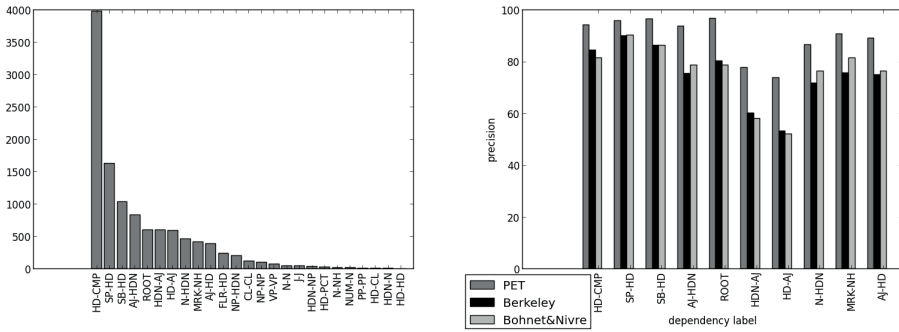


Figure 5.4: Domain: CB. (Left) Distribution of dependency labels; (right) precision of attachment of the most frequent dependency types

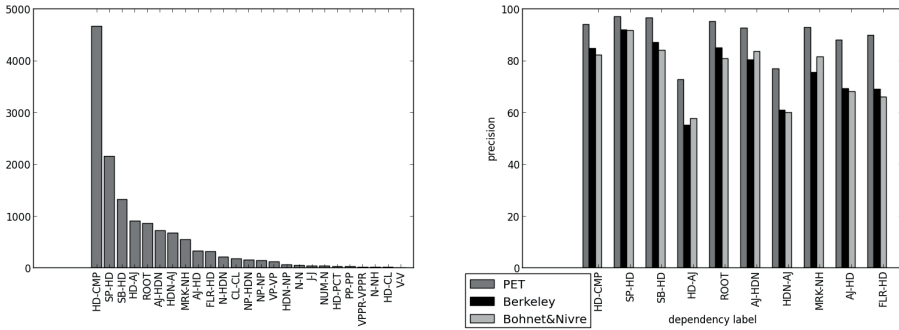


Figure 5.5: Domain: SC. (Left) Distribution and precision of dependency; (right) attachment of the most frequent dependency labels

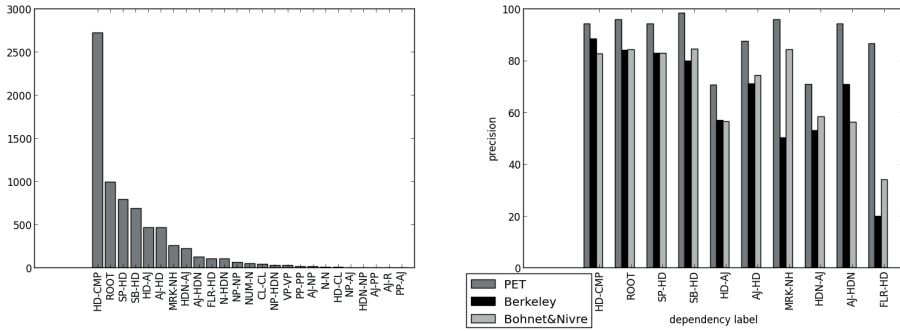


Figure 5.6: Domain: VM. (Left) Distribution of dependency labels; (right) precision of attachment of the most frequent dependency types

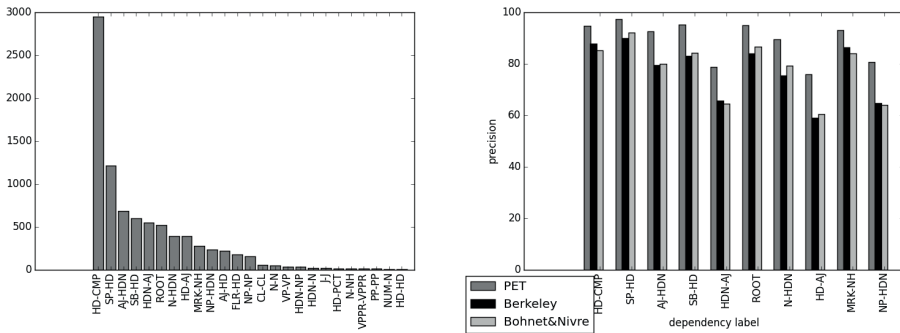


Figure 5.7: Domain: WS. (Left) Distribution of dependency labels; (right) precision of attachment of the most frequent dependency types

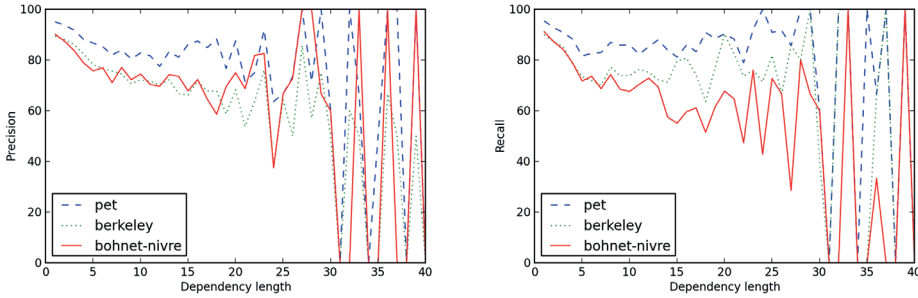


Figure 5.8: Domain: WSJ. (Left) Dependency label precision relative to predicted dependency length; (right) label recall relative to gold dependency length

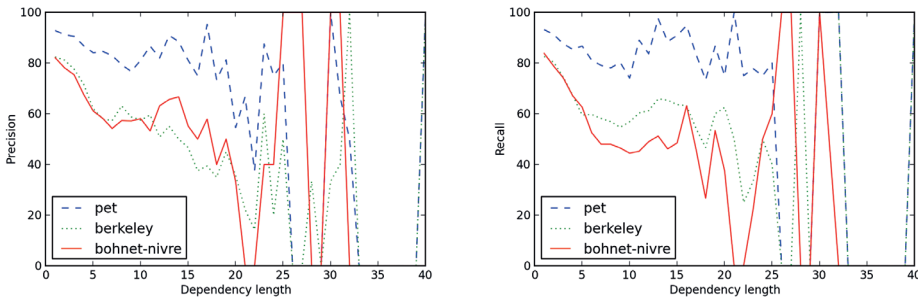


Figure 5.9: Domain: CB. (Left) Dependency label precision relative to predicted dependency length; (right) label recall relative to gold dependency length

few dependencies than SC domain although it comprises more sentences. In our experiments, we find that Berkeley and B&N have comparable precision and recall on short dependencies (up to five words), and on longer dependency relations (5-15 words) Berkeley holds a slight edge over B&N, with PET systematically outperforming the two other parsers. Although B&N has the lowest recall across domains, in terms of precision it surpasses Berkeley on dependencies spanning 10-15 words on CB and VM.

The histograms in Figures 5.13-5.17 present the distribution of PoS tags over the datasets and the distribution of LAS over different lexical categories. We observe that over all the domains, the category *x* (others) has low frequency (below 75 in general, and 1 on WS). All parsers show good performance on determiners (*d*) and complimentizers (*cm*). Nouns and verbs (*n* and *v*) are frequent on all domains and parsers generalize quite well, while conjunctions (*c*) and various types of prepositions (*p* and *pp*) are the most difficult for all three parsers. The cross-domain picture is similar to the in-domain, but we find that the difference between the accuracy of PET and the data-driven parsers on adjectives, adverbs and conjunctions is more pronounced on the out-of-domain data than on the WSJ test set.

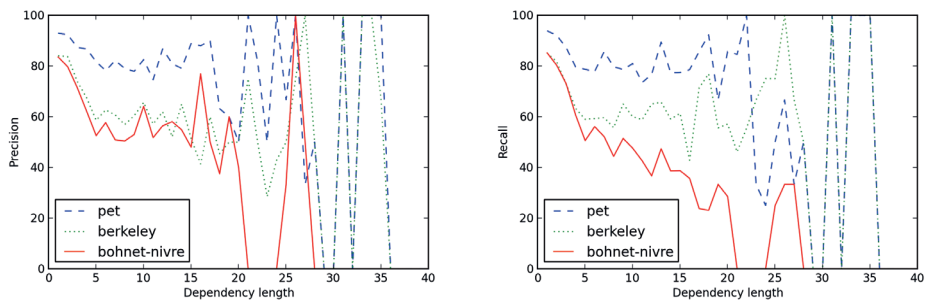


Figure 5.10: Domain: SC. (Left) Dependency label precision relative to predicted dependency length; (right) label recall relative to gold dependency length

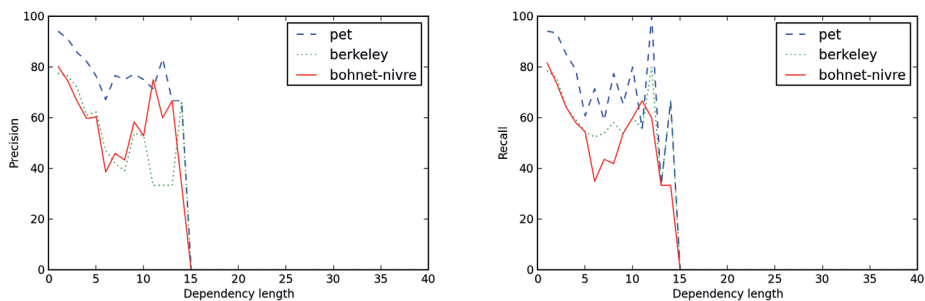


Figure 5.11: Domain: VM. (Left) Dependency label precision relative to predicted dependency length; (right) label recall relative to gold dependency length

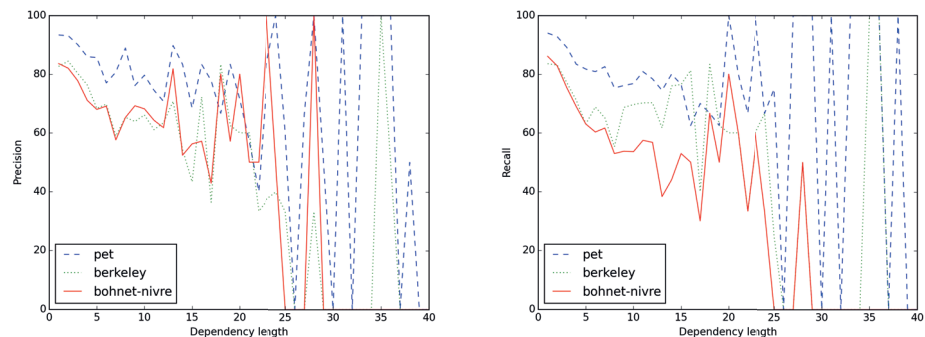
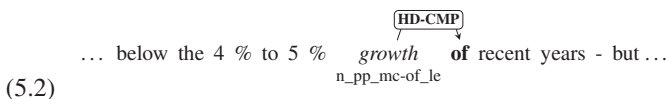
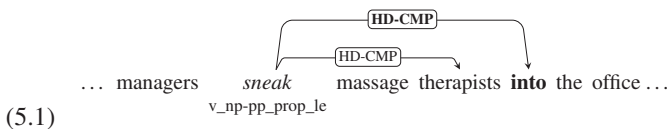


Figure 5.12: Domain: WS. (Left) Dependency label precision relative to predicted dependency length; (right) label recall relative to gold dependency length

It has been shown in the literature that coordinate structures headed by conjunctions are harder to parse for direct dependency parsers while this analysis is more expressive linguistically (Schwartz et al., 2012), therefore it is not surprising that the DT analysis of coordination is challenging for B&N. However, our results establish the same effect also for the PCFG parser and to a lesser degree for ERG_a. Nonetheless, conjunctions are among the lexical types on which PET most clearly outperforms the other two parsers with respect to their attachment and label. As it is evident from Table 5.7 which shows error rates of the parsers for the most common conjunctions “and”, “but” and “or” on the four domains, the error rates of PET do not exceed 24%, while the error rates of other parsers vary in the range 24%-34% for the lexical type *c_xp_and_le* (conjunction “and”). For many of the coordinate structures parsed correctly by ERG_a but not the other two on WSJ, we found that attachment to root constitutes the most frequent error type – indicating that clausal coordination is particularly difficult for the data-driven parsers.

Attachment of prepositional phrases is a significant and frequent source of ambiguity. ERG lexical types provide a lot more detailed analysis of prepositions than standard PTB tags, for instance distinguishing lexicalized PPs like “*in full*” among others and making explicit the difference between semantically contentful and vacuous prepositions, see below. An interesting result is that parsers have high labeled accuracy scores for some preposition classes while failing to parse the others. Table 5.8 shows four preposition types for which all three parsers make accurate predictions: *p_np_ptcl-of_le* (“*history of Linux*”)—a semantically vacuous *of*, *p_np_ptcl_le* (“*look for peace*”)—another semantically vacuous preposition, *p_np_i_le* (“*talk about friends*”)—intersective preposition, *p_np_i-reg_le* (“*starting at 75 cents*”)—preposition which constrains its complement to a non-temporal noun phrase; all four types select for noun phrase complements. For example, *p_np_ptcl-of_le* ranks among the twenty most accurate lexical categories across the board. A frequent error of the statistical parsers for the preposition type *p_np_ptcl-of_le* is attachment of the preposition to the number instead of the % symbol in phrases like “64% of”. The errors related to incorrect attachment to coordinating conjunction instead of conjunct are associated with the lexical types *p_np_ptcl_le* and *p_np_i_le*, whilst for *p_np_ptcl_le* and *p_np_i-reg_le* errors on long dependencies are most typical. PET demonstrates significantly stronger results than the data-driven parsers, which is most likely by virtue of lexical information about the argument structure of heads encoded in the grammar.

The ERG lexical rules, that function as lexical types in DT, encode valency information in the form of an ordered sequence of complements for the given type, for example *v_np_pp_prop_le* is a verb that requires two complements: a noun phrase and a prepositional phrase (see Example (5.1)).



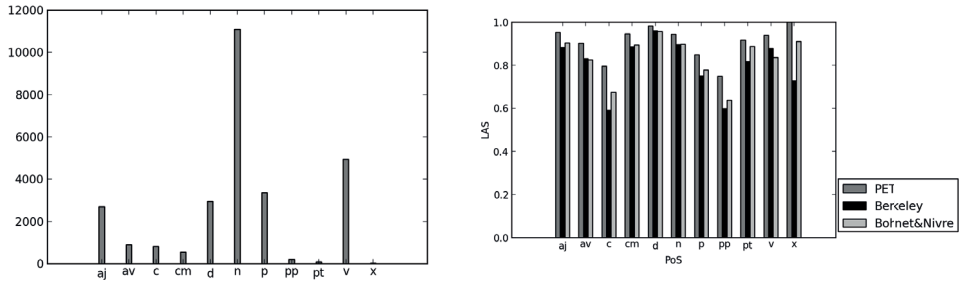


Figure 5.13: Domain: WSJ. (Left) Distribution of PoS tags; (right) accuracy of parsers for different PoS tags. v - verb, n - noun, aj - adjective, av - adverb, p - preposition, pp - prep phrase, d - determiner, c - conjunction, cm - complementizer, x - miscellaneous, pt - punctuation

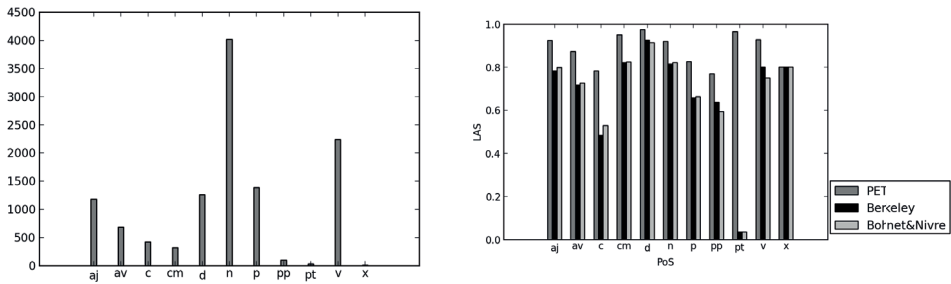


Figure 5.14: Domain: CB. (Left) Distribution of PoS tags; (right) accuracy of parsers for different PoS tags

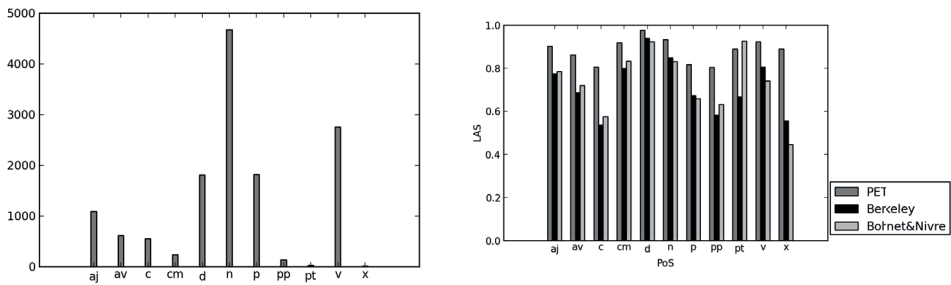


Figure 5.15: Domain: SC. (Left) Distribution of PoS tags; (right) accuracy of parsers for different PoS tags

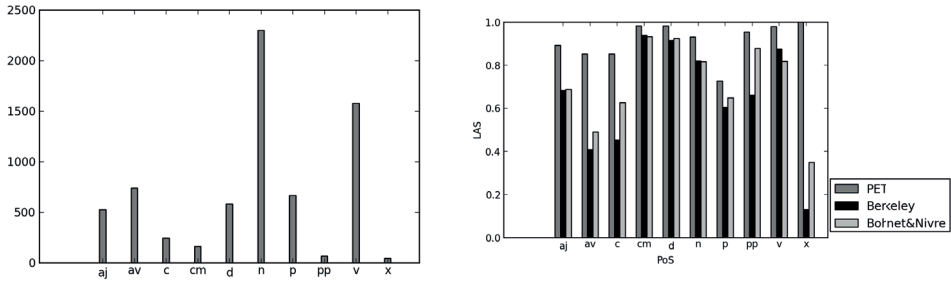


Figure 5.16: Domain: VM. (Left) Distribution of PoS tags; (right) accuracy of parsers for different PoS tags

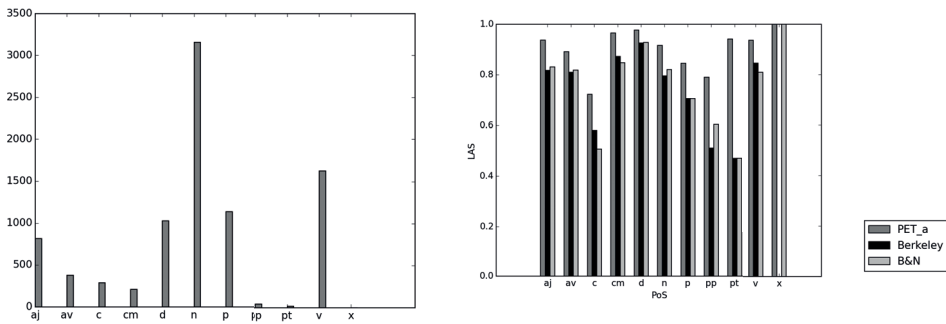


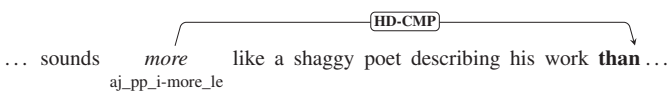
Figure 5.17: Domain: WS. (Left) Distribution of PoS tags; (right) accuracy of parsers for different PoS tags

Labeled accuracy error rate								
Lex. type	Freq.	PET	Berkeley	B&N	Freq.	PET	Berkeley	B&N
	WSJ				CB			
c_xp_and_le	521	14%	24%	24%	264	17%	31%	34%
c_xp_but_le	126	6%	8%	9%	61	7%	18%	25%
c_xp_or_le	92	12%	18%	27%	40	20%	40%	45%
	SC				VM			
c_xp_and_le	400	16%	29%	34%	177	7%	18%	17%
c_xp_but_le	69	4%	10%	12%	27	4%	7%	11%
c_xp_or_le	41	17%	29%	39%	26	19%	62%	65%
	WS							
c_xp_and_le	199	24%	32%	34%				
c_xp_but_le	12	8%	33%	17%				
c_xp_or_le	49	14%	29%	37%				

Table 5.7: Error rates of PET, Berkeley and B&N for the three frequent conjunctions (“and”, “but” and “or”) on WSJ, CB, SC and VM domains

LAS								
Lex. type	Freq.	PET	Berkeley	B&N	Freq.	PET	Berkeley	B&N
	WSJ				CB			
p_np_ptcl-of_le	358	100%	92%	94%	191	98%	90%	88%
p_np_ptcl_le	378	94%	78%	77%	174	98%	72%	70%
p_np_i_le	744	80%	71%	64%	285	76%	62%	64%
p_np_i-reg_le	707	79%	72%	76%	207	73%	62%	64%
	SC				VM			
p_np_ptcl-of_le	178	98%	92%	90%	28	100%	93%	93%
p_np_ptcl_le	224	96%	76%	71%	54	85%	63%	54%
p_np_i_le	369	78%	59%	66%	205	68%	58%	61%
p_np_i-reg_le	378	75%	63%	66%	88	74%	64%	73%
	WS							
p_np_ptcl-of_le	161	0%	2%	1%				
p_np_ptcl_le	157	1%	7%	6%				
p_np_i_le	212	12%	16%	18%				
p_np_i-reg_le	212	10%	14%	15%				

Table 5.8: Labeled accuracy scores for the four frequent lexical types for prepositions for which PET outperforms Berkeley and B&N

(5.3) 

In order to gain a better understanding of the differences observed in the analysis of prepositions, we analyzed parse errors on prepositional complements for heads of various lexical types, including most frequent verb, noun and adjunct, illustrated by Examples (5.1), (5.2)

type of head	total	ERG _a	Berkeley	B&N
aj_pp_i-more_le	20	20	19	19
aj_pp_i_le	21	21	19	19
n_pp_c-of_le	93	93	86	88
n_pp_mc-of_le	82	82	75	79
v_p-np_le	58	58	51	53
v_pp_e_le	80	78	70	54

Table 5.9: Number of total and correctly analyzed with ERG_a, Berkeley and B&N prepositional complements for various lexical types of the head—adjunct, noun and verb—on the WSJ domain

domain	total	ERG _a	Berkeley	B&N
WSJ	940	905	778	799
CB	469	446	348	354
SC	602	553	471	454
VM	164	142	113	119
WS	372	361	293	289

Table 5.10: Number of total and correct analyses of prepositional complement structures

and (5.3). Example (5.1) depicts the analysis of the predicate-argument structure of the verb (“sneak”) with a noun phrase (“massage therapists”) and a prepositional phrase (“into the office”). Both B&N and Berkeley incorrectly define the head of the phrase “into the office” as noun “therapists”, while ERG_a delivers the parse tree that corresponds to the gold standard. In Example (5.2) ERG_a correctly identifies “growth” as the head of prepositional phrase “of recent years” at the same time as B&N attaches “of” to the word “4” and Berkeley to the conjunction “but” with erroneous dependency labels. In Example (5.3), ERG_a has no problems with the prepositional complement, and B&N and Berkeley predict the proper label, but wrongly suggest attachment to the noun “work” and verb “sounds” correspondingly.

In most cases the lexical category of the head explicitly shows the requirement of a prepositional complement and taking advantage of this rule ERG_a consistently outperforms other parsers in- and cross-domain as depicted in Table 5.10.

It appears that lexical information about the argument structure of heads encoded in the grammar allows ERG_a to analyze these prepositions (in context) much more accurately.

5.7 Sanity experiments

Based on the results presented in the previous section it may be argued that the superior performance of PET is caused by a large set of complex lexical types, sparsely distributed over the corpus and the alternative tokenization approach. By design, the grammar-based parser is better adapted to handle data in this format than statistical data-driven parsers.

In this section we are going to verify our results in more conventional setup, ruling out the effects of the specifics of the ERG lexical rules and ERG-style tokenization. The goal is to

		Gaps	Lexical Types scoring punct.		PTB PoS Tags scoring punct.	
			LAS _e	UAS _e	LAS _e	UAS _e
WSJ	B&N	0+0	88.78	91.52	91.56	93.63
	ERG _e	13+9	92.38	93.53	92.38	93.53
CB	B&N	0+0	81.56	86.18	84.54	88.53
	ERG _e	8+4	90.77	92.21	90.77	92.21
SC	B&N	0+0	81.69	86.11	85.17	88.85
	ERG _e	11+0	90.13	91.86	90.13	91.86
VM	B&N	0+0	77.00	83.73	82.76	88.11
	ERG _e	10+0	91.55	93.08	91.55	93.08
WS	B&N	0+0	82.09	86.17	84.59	88.41
	ERG _e	4+0	91.61	92.62	91.61	92.62

Table 5.11: Coverage and dependency accuracies with PTB tokenization and either detailed or coarse lexical categories

confirm that the performance of PET is not caused by its bias to particulars of unification-based grammars or its internal capacities for morphological analysis guided by the ERG lexical rules. The Berkeley parser is not included in this study because its output cannot be transformed to a dependency tree with PTB tokens without substantial modifications of the original ERG derivation structure due to technical reasons. In these experiments the B&N and ERG_e parsers are run with token-splitting style that complies with the PTB tradition, first using ERG lexical rules as PoS, and then using only the standard set of PTB PoS tags.

The results are presented in Table 5.11. We observe that tokenization method indeed affects the accuracies of both parsers and the PTB approach appears to be easier: the performance of B&N improves by 2% on the in-domain data and between 1.5% and 3.9% on the out-of-domain sets; for PET the increase in scores is less drastic and is below 1% on all the domains. The difference between in-domain LAS for PET and B&N is diminished by 1.2% with the change of the tokenization model, but the qualitative picture is still the same with PET producing stronger results. The cross-domain differences in performance of PET and B&N remain extensive.

With further simplification by switching to PTB PoS tags, we find that the performance of B&N on WSJ is boosted and the gap between PET and B&N reduces to 0.82% (labeled and unlabeled attachment scores of ERG_e do not change when moving from ERG lexical types to PTB PoS because PET predicts both tag sets simultaneously, which means no parser re-training or re-running is needed and the choice of the PoS type depends only on the settings of the converter). However, ERG_e still provides 10% error reduction compared to B&N. The accuracy difference between PET and B&N remains pronounced on the out-of-domain datasets and the trend is the same as with ERG lexical types and ERG-style tokenization: while the cross-domain performance of PET is comparable with in-domain results, B&N suffers significant accuracy loss due to the domain shift. These results support the previous statements about the domain-resilience of the grammar-based parser.

In the DT format punctuation symbols are attached to the nearest tokens to the left, which makes statistical analysis of punctuation relatively simple. To eliminate this effect, we compute

		PTB PoS Tags			
		scoring punct.		not scoring punct	
		LAS	UAS	LAS	UAS
WSJ	B&N	91.56	93.63	90.45	92.80
	ERG _e	91.82	92.97	91.08	92.38
CB	B&N	84.54	88.53	83.35	87.65
	ERG _e	90.54	91.97	89.80	91.37
SC	B&N	85.17	88.85	83.56	87.62
	ERG _e	89.95	91.68	89.05	90.97
VM	B&N	82.76	88.11	79.73	86.02
	ERG _e	91.64	93.17	90.36	92.15
WS	B&N	84.59	88.41	83.20	87.46
	ERG _e	91.61	92.62	90.78	91.91

Table 5.12: Dependency accuracies computed with the eval.pl software for PTB tokenization and coarse lexical categories

scores without taking into account punctuation. The RESA software does not support dependency scoring without punctuation, for this reason we have to turn to the eval.pl software. For B&N the switch of evaluation software makes no difference when scoring punctuation because the parser provides full coverage and receives gold PTB PoS as input. For ERG_e, however, there are cases of parse failures and tokenization deviating from the gold standard where eval.pl penalizes such cases stricter than RESA, i.e. the whole sentence is counted as incorrect by the eval.pl software if there is any tokenization mismatch with the gold standard (compare Tables 5.11 and 5.12, evaluation on PTB tokens with punctuation for ERG_e). Table 5.12 contrasts evaluation with and without punctuation symbols: the in-domain difference in LAS reduces to 0.63 percentage points while the gap on the cross-domain data remains over 5 percentage points. Thus, the results of evaluation ignoring punctuation are qualitatively very similar to evaluation with punctuation and do not add any novelty to our findings.

5.8 Extrinsic evaluation on negation resolution task

Using the task of negation resolution introduced in Section 4.5, we would like to test whether the improved accuracy of the HPSG-based parser translates into better performance of the negation resolution system. As in Section 4.5, we use the UiO₂ system (Lapponi et al., 2012b), the Conan Doyle corpus and the evaluation software of Morante and Blanco (2012). We produce DT dependencies with PTB PoS tags for the NR software using PET and B&N. The Berkeley parser is not included in the experiment because UiO₂ requires PTB tokenization.

Intrinsic parser evaluation on the 91 manually annotated sentences taken from the story *Wisteria Lodge*, a sub-set of Conan Doyle development data, is shown in Table 5.13. The Conan Doyle data set represents a new domain for the grammar as it was not available before the release of the ERG in version 1212 applied in the present work. According to our expectations, the results are very similar to the cross-domain evaluation in Table 5.11.

	Gaps	TA _c	LAS _c	UAS _c
B&N	0+0	92.24	83.92	87.92
ERG _a	0+0	96.36	92.54	93.84
ERG _e	0+3	94.21	89.22	90.57

Table 5.13: Parse failures and token mismatches (‘gaps’), and tagging and dependency accuracy on the sub-set of the Conan Doyle development data

	ERG _a			ERG _e		
	Train	Dev	Test	Train	Dev	Test
total # sentences	3644	787	1089	3644	787	1089
% Coverage	89.96	91.11	87.42	81.64	83.99	79.98
% Gold tokenization	89.24	91.11	86.23	80.98	83.86	78.88
% Fall-back	10.76	8.89	13.77	19.02	16.14	21.12

Table 5.14: PET coverage on Conan Doyle training, development and test sets

Unlike B&N, PET does not deliver complete coverage because of parse failures (especially the ERG_e variant due to excessive pruning of the search space) and tokenization mismatches with the gold standard (since processing starts from raw strings). For example, 26 sentences from the training set parsed with ERG_a differ from the gold sentences in tokenization of the word “cannot”; for ERG_e there are 24 such sentences. For the sentences that are not parsed by PET, we fall back to the analyses of B&N. Table 5.14 shows the statistics that describe the above mentioned issues: the coverage of PET, the proportion of sentences that received analysis matching the gold tokenization and the proportion of sentences for which the fall-back to the B&N parses is enabled, i.e. for ERG_a 89.24% of the training set consists of the PET analyses and 10.76% of the B&N analyses.

Tables 5.15 and 5.16 show the results for ERG_a, ERG_e and B&N with DT dependencies on the development and test sets respectively. We include the scores from Lapponi et al. (2012b) using the Malt parser with Stanford Basic dependencies for comparison. Somewhat unexpectedly, the performance contrasts observed in intrinsic parser evaluation in Table 5.6 are not reflected in the results of the extrinsic evaluation. The paired sign test shows that none of the differences in experiments with the negation resolution system are statistically significant.

One hypothesis, explaining the results is that the accuracies of the parsers on the part of the Conan Doyle domain that was not manually annotated are not significantly different, unlike on the other domains. For testing this proposition, a manual HPSG annotation of the entire Conan Doyle corpus is required. Another hypothesis that can explain the current picture is that negation resolution in our setup is not as sensitive to parser accuracy as the other tasks mentioned in the related work section, i.e. answer extraction (Mollá and Hutchinson, 2003), protein-protein interaction extraction (Miyao et al., 2008) and event extraction (Miwa et al., 2010). In order to develop this hypothesis, we could choose another downstream application relying on syntactic dependencies, such as semantic dependency parsing (Oepen et al., 2014).

	ERG _a	ERG _e	B&N	Malt
Cues	100	100	100	100
Scopes	80.00	80.85	80.43	80.00
Scope tokens	86.65	88.10	88.13	85.75
Negated	75.73	73.33	73.33	80.55
Global	64.31	63.24	65.89	66.41
CNS	45.83	44.44	47.92	-

Table 5.15: Performance of the negation resolution system UiO₂ on the development set with gold cues on the Conan Doyle corpus from the 2012*SEM shared task using dependency features from the analyses of ERG_a, ERG_e and B&N with DT dependencies in contrast with the analyses of the Malt parser with SB dependencies from Lapponi et al. (2012b)

	ERG _a	ERG _e	B&N	Malt
Cues	91.31	91.31	91.31	91.31
Scopes	73.52	74.83	75.40	72.39
Scope tokens	83.90	83.61	85.09	82.20
Negated	61.29	60.95	60.44	61.79
Global	53.73	55.53	55.17	54.82
CNS	40.43	41.28	41.28	-

Table 5.16: Performance of the negation resolution system UiO₂ on the test set with gold cues on the Conan Doyle corpus from the 2012*SEM shared task using dependency features from the analyses of ERG_a, ERG_e and B&N with DT dependencies in contrast with the analyses of the Malt parser with SB dependencies from Lapponi et al. (2012b)

5.9 Extrinsic evaluation on semantic dependency parsing

Since the results of the extrinsic evaluation on negation resolution have not confirmed our expectations, we are going to explore a second task, namely broad-coverage semantic dependency parsing (SDP), a shared competition from SemEval 2014 (Oepen et al., 2014). The best-performing system in the open track of the contest, called Priberam Martins and Almeida (2014), is based on a feature-rich linear model that can take advantage of information from the syntactic dependency parser. In the following, we will investigate whether syntactic dependency features provided by the grammar-based system facilitate more accurate semantic parsing than features delivered by the data-driven tools.

Broad-Coverage Semantic Dependency Parsing

By the definition from SemEval 2014, Broad-Coverage Semantic Dependency Parsing is the problem of recovering sentence-internal predicate—argument relationships for all content words. In contrast to syntactic parsing which targets the grammatical structure of a sentence, semantic parsing aims to recover semantics, and while syntactic analysis is usually limited to tree structure representations, semantic parsing applies to more general graph processing. Semantic dependency parsing is to some extent related to semantic role labeling (Gildea and Jurafsky, 2002). However the latter task is usually restricted to argument identification and argument labeling of nominal and verbal predicates, while semantic parsers aim at recovering semantic dependencies between all content words in a sentence. Another difference between the tasks is that SDP in SemEval 2014 is limited to structural analysis by design and it does not address the problem of predicate sense disambiguation.

The SDP 2014 dataset consists of sections 0-21 of the WSJ corpus annotated with the three target representations which have been presented in Chapter 3: DM, EP and PT aligned on the sentence and token levels. WSJ sentences that (a) lack gold-standard annotation in one of the formats, or (b) do not exactly match across representations on the token level, or (c) are analyzed with cyclic graph structures,—are not included in the set. We have used the resource in our contrasting analyses in Section 3.5.

The task is organized into two tracks: systems in the *closed* track were trained only on the data distributed by the task organizers while the systems in the *open* track could use additional resources. We are, therefore, only interested in the latter track. In the open track of the SemEval 2014 shared task, participants were offered companion files containing Stanford Basic syntactic dependencies produced by B&N. The evaluation measures are labeled precision (LP), labeled recall (LR), labeled F1 (LF), and labeled exact match (LM) with respect to predicted triples “predicate - dependency label - argument”.

The Priberam system of Martins and Almeida (2014) which relies on a model with second-order features and decoding with dual decomposition, was ranked first in the open challenge, and achieved the second highest score in the closed track. By virtue of syntactic features extracted from the output of the syntactic dependency parser, Priberam attains a consistent improvement of around 1% in LF for all three semantic dependency formats. We have chosen this system for our extrinsic evaluation.

	ERG _a	ERG _e	B&N	SemEval'14
LP	90.88	90.77	88.96	90.23
LR	89.86	89.67	88.10	88.11
LF	90.37	90.22	88.53	89.16
LM	32.42	32.64	29.75	26.85

Table 5.17: SemEval 2014 open track results of the Priberam system on DM

	ERG _a	ERG _e	B&N	SemEval'14
LP	92.04	92.19	91.91	92.56
LR	89.67	89.89	89.63	90.97
LF	90.84	91.03	90.75	91.76
LM	31.38	30.93	32.86	37.83

Table 5.18: SemEval 2014 open track results of the Priberam system on PAS

Parser comparison

In the following, we compare the quality of syntactic features produced by ERG_a, ERG_e and B&N for the semantic parsing with Priberam. Using these three parsers we prepare companion data with DT bilexical dependencies. Out of the 1348 sentences from the SDP test set, ERG_a and ERG_e failed to deliver analysis for 11 and 24 sentences respectively, and, as in the previous extrinsic evaluation on negation resolution in Section 5.8, we borrowed the missing data from output of B&N which achieves complete coverage.

Tables 5.17, 5.18 and 5.19 present the results for DM, PAS and PCEDT respectively. For comparison, we include the results of Priberam from the SemEval 2014 shared task using a companion file generated by the task organizers using B&N with Stanford Basic dependencies. We observe that the correlation of syntactic and semantic dependency schemes facilitates improved semantic parsing: the best results for the DM scheme are obtained with a companion file with DT dependencies constructed by ERG_a, the top results for PAS—with the SB dependencies from the B&N analyses, and for PCEDT all results are significantly lower than the Priberam results in all setups. Both DT and DM are derived from ERG, and SB and PAS are related because the SB dependencies for training B&N are generated from phrase structure parses of PTB while PAS are derived from the HPSG-style annotation automatically produced from the original bracketing annotations of Penn Treebank. In other words, DT and DM originate from DeepBank and SB and PAS originate from PTB.

The results of the evaluation furthermore show that semantic parsing is clearly more sensitive to parser performance than negation resolution, especially taking in account that from Table 5.11 the maximum in-domain difference between ERG_e and B&N in LAS is 0.82% when using PTB tokenization with PTB PoS tags.

	ERG _a	ERG _e	B&N	SemEval'14
LP	79.62	79.94	79.42	80.14
LR	75.67	75.82	75.45	75.79
LF	77.59	77.82	77.38	77.90
LM	11.05	11.20	10.98	10.68

Table 5.19: SemEval 2014 open track results of the Priberam system on PCEDT

5.10 Reflections on the SDP 2014 results

In this section we discuss some of the outcomes of the Task 8 Broad-Coverage Semantic Dependency Parsing (SDP; Oepen et al. (2014)) at SemEval 2014 which we have already introduced in the previous section. The official results of the task are in parallel with the observations made in the current chapter: in the present work we have shown that the grammar-based parser has higher accuracies than statistical analyzers on the syntactic format DT and that grammar-based parsers are very competitive with the purely statistical systems on the semantic annotation formats DM and EP (Miyao et al., 2014). The task can be also viewed as an intrinsic evaluation of parsers on the three semantic dependency formats DM, EP and PT that were structurally compared in Chapter 3.

One of the objectives of the shared task is to encourage engineering efforts in semantic graph-based dependency parsing in contrast to syntactic tree-oriented processing. Tree structures are not well-suited for semantic parsing because in semantic analysis some nodes have several semantic heads while other nodes do not carry semantic meaning on their own and should be omitted from the graph. These requirements contradict tree constraints which motivates more general processing of labeled directed graphs.

In the final results there are accuracies for the 6 different systems in the closed track and 6 different systems in the open track. We are most interested in the open track results because these include evaluations of the grammar-based parsers PET and Enju⁵.

The systems that submitted results to the open track are called Priberam, CMU, Turku, Potsdam, Alpage and In-House. For some systems there were two runs submitted. Priberam (Martins and Almeida, 2014) is a linear model with second-order features (extension of the TurboParser (Martins et al., 2013)); CMU (Thomson et al., 2014) is a feature-rich arc-factored discriminative model; Turku (Kanerva et al., 2014) is a pipeline system of three support vector machine classifiers for selecting dependencies, predicting dependency labels and selecting top nodes; Potsdam (Agić and Koller, 2014) are bidirectional graph-to-tree transformations combined with syntactic dependency parsing; Alpage (Ribeyre et al., 2014) is two transition-based dependency parsers with additional actions to handle graphs; and In-House (Miyao et al., 2014) is an ensemble of parsers developed in parallel to the creation of the dependency representations: the PET parser with ERG 1212 grammar for DM, the Enju parser for EP and the Treex system for PT. There is no specialized grammar-based parser for PT and the Treex system combines data-driven dependency parsing with tectogrammatical conversion, therefore we will

⁵The “in-house” parsers are trained with knowledge derived from additional resources which was the reason why their runs were not submitted in the closed track.

Team	Run	DM				EP			
		LP	LR	LF	LM	LP	LR	LF	LM
Priberam	1	0.9023	0.8811	0.8916	0.2685	0.9256	0.9097	0.9176	0.3783
CMU	1	0.8446	0.8348	0.8397	0.0875	0.9078	0.8851	0.8963	0.2604
Turku	2	0.8094	0.8214	0.8153	0.0823	0.8733	0.8776	0.8754	0.1721
Turku	1	0.7989	0.8245	0.8115	0.0757	0.8742	0.8801	0.8771	0.1780
Potsdam	2	0.8132	0.8091	0.8111	0.0905	0.8941	0.8261	0.8588	0.0749
Alpage	1	0.8346	0.7955	0.8146	0.1076	0.8723	0.8282	0.8497	0.1543
Alpage	2	0.8577	0.7446	0.7971	0.0660	0.8860	0.8276	0.8558	0.1091
Potsdam	1	0.8454	0.7380	0.7880	0.0415	0.8972	0.7508	0.8175	0.0178
In-House	1	0.9258	0.9234	0.9246	0.4807	0.9209	0.9202	0.9206	0.4384

Table 5.20: Labeled Dependencies Including TOP Nodes

omit the discussion of parsing results on PT, since in this chapter we are focused on contrasting grammar-based parsing with purely statistical parsing.

Tables 5.20 and 5.21⁶ summarize labeled and unlabeled scores correspondingly on the DM and EP representations for the systems submitted to the open track of the shared task. In-House system has the highest labeled and unlabeled scores except LP and UP on EP where its results are very close to the top Priberam values. There is a possible positive bias towards the In-House ensemble in the testing data, similar to the bias towards the ERG parser on DT dependencies in previous sections: WSJ sentences that lack gold-standard analyzes in DeepBank and Enju treebanks are excluded from the SDP data. In addition, sentences that are analyzed with cyclic graph structures in DM are also filtered out. Despite this fact, ERG and Enju parsers show very competitive results for DM and EP, especially in terms of exact match (Miyao et al., 2014).

The direct head-to-head parser comparison on syntactic (discussed in the previous sections of the present work) and semantic (SDP 2014) billexical dependencies demonstrates competitiveness of grammar-based approaches parsing. For the Enju parser comparisons to other systems have been carried out in the previous work of Miyao et al. (2007) and Bender et al. (2011) and are in line with our conclusions. The strength of the grammar-based parsers can be explained by the depth of the linguistic information expressed in machine-readable format by the grammar writers that is exploited during parsing.

5.11 Summary

In this chapter we compared the HPSG-driven parser PET with data-driven syntactic analyzers using billexical dependencies as the basis for comparison. Our intrinsic evaluation on syntactic dependencies showed superior accuracy and resilience to domain and genre variation for the ERG parser, which corresponds to the findings of Plank and van Noord (2010) for Dutch language. The efficiency of the grammar-driven parser achieves the levels of the best direct dependency parser. However, unlike the data-driven dependency and phrase structure grammar

⁶<http://svn.delph-in.net/sdp/public/2014/results.tgz> Accessed: 22 August 2014.

Team	Run	DM				EP			
		UP	UR	UF	UM	UP	UR	UF	UM
Priberam	1	0.9141	0.8926	0.9032	0.2990	0.9362	0.9202	0.9281	0.3924
CMU	1	0.8603	0.8503	0.8553	0.0979	0.9179	0.8950	0.9063	0.2715
Turku	2	0.8287	0.8410	0.8348	0.0964	0.8876	0.8918	0.8897	0.1803
Turku	1	0.8182	0.8444	0.8311	0.0838	0.8885	0.8945	0.8915	0.1869
In-House	1	0.9361	0.9337	0.9349	0.5230	0.9320	0.9314	0.9317	0.4429
Alpage	2	0.8819	0.7656	0.8197	0.0779	0.9005	0.8412	0.8699	0.1150
Alpage	1	0.8574	0.8172	0.8368	0.1194	0.8895	0.8445	0.8664	0.1662
Potsdam	2	0.8337	0.8295	0.8316	0.1031	0.9078	0.8387	0.8719	0.0779
Potsdam	1	0.8643	0.7545	0.8057	0.0497	0.9099	0.7614	0.8291	0.0208

Table 5.21: Unlabeled Dependencies Including TOP Nodes

parsers, PET has partial coverage, and in most of this work we had to ignore some 15% of the original data due to the absence of the gold-standard annotations, traditionally produced with the ERG and PET. In practical applications, this flaw can be overcome by falling back on outputs of another parser for sentences lacking analysis, as we did in our extrinsic evaluation. The better performance of the grammar-driven parser did not translate to the improved results of the negation resolution task, chosen as the first task for extrinsic evaluation, while evaluation on semantic dependency parsing, selected as the second task for extrinsic evaluation, correlates with the results of the in-domain intrinsic parser comparison. In future work it would be interesting to include more PCFG and transition-based dependency parsers in our comparison and test parser outputs in other downstream applications, sensitive to results of syntactic analysis.

Similarly to our experiments, the shared task on Broad-Coverage Semantic Dependency Parsing (SDP; Oepen et al. (2014)) can be seen as an intrinsic cross-paradigm evaluation of parsers on semantic dependencies. Likewise to observations from our study, the tentative conclusion is that linguistically rich parsing achieves higher accuracy levels.

Chapter 6

Parser combination

A recurrent issue in grammar-based parsing concerns efficiency. In this chapter we seek to prune the search space of the ERG parser based on high-confidence dependency predictions. The goal of these experiments is to improve parser efficiency and in order to achieve it we carry out a careful analysis of trade-offs across the dimensions of efficiency, coverage and accuracy. Our hypothesis is that partly conflicting requirements of these three dimensions can be jointly addressed and optimized by parser combination.

6.1 Introduction

The main criteria for assessment of parser performance are efficiency, coverage and accuracy. A good balance of the three is a challenge for grammar-based parsing. In order to investigate the possibility of improving and balancing efficiency, coverage and accuracy of ERG parsing we experiment with parser combination, using bilexical dependencies to constrain the PET parser. At the outset of our experimentation, we considered two different approaches to parser combination which we will dub “*loose*” and “*tight*” integration.

With “loose” methods no modifications are introduced in the parsing mechanisms, but rather the output of the dependency parser is employed as features to improve the stochastic parse ranker. Kim et al. (2012) proposed to incorporate dependency features into a CCG reranker to reorder n-best outputs of a CCG parser. Another method, implemented in Marimon et al. (2014) for an HPSG parser is to compare the analysis delivered by the dependency parser and the one chosen by the parse ranker, and then accept the automatically proposed analysis only if both are identical.

With the “tight” methods the output of dependency parsers is used to modify the parsing process of the grammar-based parser. Sagae et al. (2007) use bilexical dependencies to constrain the application of wide-coverage HPSG rules during parsing. The authors distinguish “*soft*” and “*hard*” constraints. When dependencies are applied as “hard” constraints the grammar-based parser is forced to produce an analysis that strictly conforms to the output of the dependency parser. When dependencies are considered as “soft” constraints the log-likelihood of the partial parse trees is penalized if the application of HPSG rules results in structures that do not correspond to the trees generated by the dependency parser.

The “loose” mechanism of parser combination does not interact with the parsing process

but rather re-shuffles parser outputs. Choosing that, we could only hope for some accuracy improvements, but not any significant efficiency benefits. We therefore opted for the “tight” method because it offers integration on a deeper level, affecting the parsing process directly, thus it is promising for efficiency advances and possibly also modest accuracy gains. Applying dependencies as “hard” constraints would mean performing extensive pruning which might lead to significant loss of coverage as a side effect, therefore a safer alternative is using dependencies as “soft” constraints to guide the parsing process.

In the following we will first review some previous work aimed at improving various aspects of grammar-based parsing before we go on to describe our own experiments aimed at improving HPSG parsing using a “tight” method of parser combination where DT dependency analyses are used as soft constraints during parsing.

6.2 Related work

In the current section we provide a brief overview of several approaches to improve efficiency, coverage and accuracy of grammar-based parsing. In focus of our discussion are lexicalized grammars (and parsing systems) that we have repeatedly mentioned in previous chapters: LFG (Kaplan and Bresnan, 1982) with the XLE system (Maxwell and Kaplan, 1993), HPSG (Pollard and Sag, 1994) with the PET (Callmeier, 2000), Alpino (Malouf and van Noord, 2004) and Enju (Miyao and Tsujii, 2005) parsers, and CCG (Steedman, 2000) with the C&C syntactic analyzer (Clark and Curran, 2007b).

6.2.1 Efficiency

Large-scale grammars involve rich formalisms and often encounter an extensive search space in selecting the correct analysis for a given utterance. Parsing is complicated by the ambiguity of natural language, in particular local ambiguities: each substring of an utterance can receive many different interpretations when analyzed without broader context. Theoretically, there is no polynomial time complexity upper-bound for unification-based parsing (Carroll, 1993)[p. 146]. In practice most parsing time is indeed spent on unification operations, however existing algorithms for parsing large feature structures usually have linear-time complexity. Another important factor is related to exact inference: The HPSG parser PET computes the complete forest and determines the globally correct n-best list of analysis, which is computationally a lot more expensive than the approximate inference methods employed in most phrase structure and dependency parsers.

Most of the techniques proposed to overcome problems of efficiency are based on reducing the search space using filtering and pruning techniques and typically lead to approximate inference. A common filtering technique is context-free grammar approximation of the HPSG representations called *CFG filtering* (Harbusch, 1990; Torisawa et al., 2000; Kiefer and Krieger, 2000) and *PCFG filtering* (Zhang and Krieger, 2011). Partial trees that are non-parseable by the approximating grammar formalism are eliminated and parsing speed therefore improves because the parser avoids unnecessary unification operations. Context-free grammar approximations can be induced with grammar-driven and corpus-driven approaches. In the grammar-driven implementation CFG rules are compiled directly from the HPSG grammar (Kiefer and

Krieger, 2004) whereas in the corpus-driven implementation CFG rules are derived from a tree-bank (Krieger, 2007). The grammar-driven method, however, is intractable in practice since it produces billions of CFG productions (Zhang and Krieger, 2011). The corpus-driven method operates by acquisition of the CFG categories and rules from the derivation trees and estimation of probabilities with naïve maximal likelihood. A disadvantage of the corpus-driven approach is unsoundness: sentences that the grammar would have accepted may be rejected if certain valid CFG rules do not occur in the corpus.

In the *beam thresholding* approach (Ninomiya et al., 2005) edges from the chart cells are pruned during parsing by tracking only n -best parses according to their figure of merit. In the *C-structure pruning* method for LFG (Cahill et al., 2008) subtrees with probabilities below a certain threshold at a particular cell in the chart are pruned based on a stochastic CFG model. Cramer and Zhang (2010) address the problem of efficiency of the PET parser by restricting the search space using an approach similar to C-structure pruning for LFG: setting a maximum number of tasks per chart cell calculating the priorities of the tasks based on a PCFG model.

Another group of methods imposes linguistically motivated local constraints on lexical items in order to improve efficiency. In *supertagging* (Bangalore and Joshi, 1999) lexical entries are assigned to words in order to encode constraints on how a word can be combined with other words and phrases (Matsuzaki et al., 2007) and in *übertagging* (Dridan, 2013a) supertagging is performed over ambiguous tokenization (see Sections 2.2.2 and 5.3 for more details).

Finally, maximum parsing time can be restricted to avoid long processing time. *Skimming* (Kaplan et al., 2004) in LFG parsing with XLE denotes an interruption of the processing of a subtree when the specified amount of memory or time spent exceeds a threshold that can be tuned by the user-specific parameters. This mechanism helps to avoid timeouts and memory exhaustion and guarantees parsing in polynomial time.

6.2.2 Coverage

Grammar engineers constantly face the challenge of ensuring coverage, i.e. enabling the processing of unseen constructions and ungrammatical texts. Nowadays parsing with hand-written precision-oriented linguistic grammars does not achieve complete coverage and occasionally fails to process longer sentences. There are two main reasons for the lack of analyses: (1) the parser exhausts its resources within given limitations on memory and space; (2) an appropriate analysis is not found among the unpacked solutions from the forest.

In the second case the correct analysis is not in the forest because some lexical items are not attested in the lexicon, certain constructions are missing from the grammar or because an input is ungrammatical or fragmented. Ungrammatical input might contain typos, word and phrase repetitions. Some constructions might be missing from the grammar because they are unusual or have not been encountered by grammar developers in the corpora that was used for grammar development. For parsing with ERG there is a distinction between “raw” and “validated” coverage, where the former term conveys percentage of sentences that are assigned an analysis by the parser and the latter term means the percentage of HPSG analyses that were produced by the parser and manually verified by multiple annotators. ERG typically misses some 10% of “validated” coverage.

Among known methods for improving the coverage of a grammar-based parser are partial

parsing and robustness rules.

Partial parsing (Kiefer et al., 1999) is a technique of building up as much of the analysis as possible and then stitching chunk parses together. In XLE, a system for parsing and generating LFGs, issues of incomplete coverage are handled by a fragment or chunk parsing. The standard LFG is augmented with a “fragment grammar” so that the input can be analyzed as a sequence of chunks. The best partial parse is the one that is constituted of the least number of chunks, e.g. if a string can be analyzed as two NPs and a VP or as one NP and an S, the NP-S version is chosen (Riezler et al., 2002). The rules in LFG are ranked based on optimality theory and rules handling fragmented and ungrammatical structures are dispreferred. Partial parsing is also implemented in the Dutch Alpino system (van Noord, 2001). Zhang et al. (2007a) experimented with integration of partial parsing into the PET parser with three different partial parse disambiguation models in order to increase parser coverage. A partial parse in this setup is a set of passive edges licensed by the grammar with non-overlapping spans that together cover all tokens of the input sentence. Zhang et al. (2007a) implement two models based on the maximized conditional probability of the partial parse modeled similarly to the maximum entropy models for full parse selection and compare them to the model based on the shortest path approach with heuristic weights in which it is assumed that the best partial parse is the one that has the shortest path (smallest summed weight) from the start vertex to the end vertex.

Cramer and Zhang (2010) proposed to add to the grammar a small number of radically overgenerating *robustness rules* that can tackle extra-grammatical sentences. To increase the robustness of the PET parser, Cramer and Zhang (2010) added a small set of overgenerating rules to the HPSG grammar for German that can deliver analyses for sentences that cannot be processed by the original grammar. The authors argue that robustness rules are superior to partial parsing proposed in Zhang et al. (2007a) since the former theoretically should fill in the gaps both in the lower and higher areas of the chart keeping the damage local when possible while the latter combines partial analyses only at the top level.

6.2.3 Accuracy

The accuracy of automatic grammar-based analysis concerns the linguistic adequacy of the syntactic trees output by the parser. As with efficiency, the problem of accuracy is closely related to the ambiguity of natural languages. Grammatically sound analyses are not necessarily correct, therefore statistical disambiguation models that emulate the structural preferences inherent to a particular language are required to tackle the challenge.

Several methods, such as symbolic approaches, parse disambiguation with statistical models, supertagging and parser combination have been proposed in prior work to ensure sufficient level of accuracy for grammar-based parsing.

In the *symbolic approach* new lexical items and rules are added to the grammar to restrict the possible space of ambiguity. This method requires intensive human effort in grammar engineering. Advanced methodologies and software such as LinGO Grammar Matrix (Bender et al., 2002, 2010) and CLIMB (Fokkens, 2011) have been proposed to facilitate manual grammar development. The LinGO Grammar Matrix customization system is a service for multilingual grammar development that comprises a core HPSG grammar covering universal linguistic phenomena and a set of libraries providing analyses for phenomena that vary across languages.

CLIMB (Comparative Libraries of Implementations with Matrix Basis) is a system for multi-lingual grammar engineering based on metagrammar development which gives grammar developers an opportunity to store competing analyses for the same linguistic phenomena and test the impact of these alternatives as the grammar grows.

The method of *parse disambiguation with statistical models* described in Toutanova et al. (2005) conveys an implementation of models that attempt to select the correct analysis for a sentence based on statistics. Toutanova et al. (2005) build discriminative and generative models over derivation trees of ERG analyses and enrich them with semantic and lexical information which facilitate significant accuracy improvement.

Supertagging (Bangalore and Joshi, 1999) is a tagging process where lexical entries assigned to the tokens of a sentence encode constraints about how a word can be combined with neighboring words and phrases. Although this method is primarily used to increase the efficiency of grammar-based parsers, there is a number of studies showing that supertagging can also facilitate gains in accuracy as well (Prins and van Noord, 2003; Foth and Menzel, 2006; Ninomiya et al., 2007). The known trade-off of advancing efficiency and accuracy with supertagging is reduced coverage.

Parser combination techniques (Henderson and Brill, 1999) encompass a range of methods for combining parsers in order to achieve improved performance. Although parser combination methods, such as PCFG filtering, are usually aimed at boosting the speed of grammar-based analyses, Sagae et al. (2007) demonstrated 1% absolute improvement in accuracy of HPSG parsing using soft dependency constraints. We will now examine in some detail previous work on parser combination and relate this work to our goal of improving efficiency of the HPSG parser.

6.2.4 Parser combination for improved efficiency and accuracy

Many (but not all) of the above methods for enhanced efficiency, coverage and accuracy have already been adapted to parsing with the ERG. Among the methods that have not yet been attempted with ERG before is parser combination. A variety of parser combination methods proposed in previous work are presented in Section 2.3. In the current chapter we experiment with parser integration in the spirit of Sagae et al. (2007), carrying-out a study of effects of dependency constraints across the three dimensions of efficiency, coverage and accuracy.

As we recall from Section 2.3, Sagae et al. (2007) applied dependency constraints generated by syntactic dependency parsers to the HPSG parser Enju (Miyao and Tsujii, 2005). The baseline in their experiments is set by running the HPSG parser on the development data without dependency constraints. The training data for the dependency parser is prepared from the HPSG treebank (Miyao et al., 2004) generated from the Penn Treebank: each sentence from the HPSG treebank is first converted to CFG by removing long-distance dependencies and then transformed to a dependency representation using a head-percolation table (Collins, 1999). The dependency parser is trained and run before HPSG parsing begins. The mechanism of HPSG parsing is modified so that before application of each HPSG construction, such as head-complement, the Enju parser identifies the corresponding dependency and checks whether this dependency is also delivered by the dependency parser. As mentioned above, the authors realize two types of constraints: *hard*, requiring the HPSG output to be strictly consistent with

dependency constraints, and *soft*, penalizing in terms of log-likelihood any inconsistency of partial parse trees created by the application of the HPSG constructions. When the HPSG structure is inconsistent with n dependencies the log-probability of the parse tree is reduced by $n\alpha$, where α is a meta parameter tuned to maximize the accuracy on the development set. The authors experiment with the following constraints for the search space of the HPSG parser:

- all the dependencies produced by the left-to-right shift-reduce SVM parser implementing the deterministic dependency parsing approach of Nivre and Scholz (2004);
- only dependencies produced by all the parsers from the parse ensemble of the left-to-right shift-reduce SVM parser and the MST parser (McDonald et al., 2005b);
- only dependencies produced by all the parsers from the parse ensemble of the left-to-right shift-reduce SVM parser, right-to-left shift-reduce SVM parser and the MST parser.

Our work is inspired by Sagae et al. (2007), as we are also using bilexical dependencies to constrain an HPSG parser and we also experiment with parser ensembles with the goal of selecting only high-quality bilexical dependencies. Nevertheless, several aspects of the present work are considerably different: first of all, we are using labeled dependencies while Sagae et al. (2007) worked with unlabeled dependency constraints; secondly, we introduce a systematic experimental study of a more comprehensive range of techniques for the selection of high-confidence dependencies; thirdly, we use a broader range of statistical dependency parsers; fourthly, we closely examine the effects of parser combination not only on accuracy and efficiency but also on coverage since the PET parser does not always deliver analysis for all the sentences in the dataset; and finally, we investigate whether results achieved on the in-domain test data translate to new domains.

6.3 Hypothesis testing

When comparing various parser integration setups to the baseline, we will use hypothesis testing to determine if the difference between a given configuration and the baseline is significant. Sjøgaard et al. (2014) show that the current practice of statistical significance testing in Natural Language Processing yields unreliable results, and the authors suggest to use a smaller significance level ($\alpha = 0.0025$ instead of commonly used $\alpha = 0.05$) and compare the systems on different datasets across available system parameters. As in Chapter 4, we establish a significance level of $\alpha = 0.001$, and compare various parser combination setups to the baseline model with respect to exact match and coverage across domains. We use three tests to verify the results of one another, though even when all the tests agree, we should accept the output with some grain of salt.

We will follow the methodology for exact match statistical significance testing explained in Vellidal (2009)[p. 139].

Exact match measures the percentage of sentences with the parse tree exactly matching the gold standard, and coverage measures the percentage of parsed sentences, e.g. sentences that were assigned at least one analysis. Exact match for an individual sentence is “1” if the parse tree matches the gold, and “0” otherwise. In case a sentence is assigned several analyses, the

score is discounted proportionally. Coverage for an individual sentence is “1” if a sentence is parsed, and “0” otherwise. The null hypothesis is that the difference in performance of a certain setup from the baseline is due to chance.

Comparing the performance of a setup to the baseline we will be looking at paired differences in scores, disregarding cases where the difference is zero. If the test set is comprised of m sentences, we have to compute the differences $a_1 - b_1, \dots, a_m - b_m$, where a_i represents a score (exact match or coverage) of the i th sentence for a setup in question (A) and b_i stands for a score for the baseline (B) analysis for the i th sentence. We will ignore cases where the difference is zero, which leaves us with a possibly reduced list of differences of size n . The remaining differences in our list will be in most cases either $+1$, or -1 ; except for the cases when the difference in exact match is a fractional number due to multiple analyses assigned to a sentence.

Paired sign test This list of list of differences can be seen as a sequence of Bernoulli trials with two possible outcomes—a positive value of a difference, and a negative value of a difference—with an underlying binomial distribution. The null hypothesis is that the performance of a given setup is not significantly different from the baseline, i.e. the probability of success in a Bernoulli experiment is $p = 0.5$ meaning that there is equal chance for the difference $a_i - b_i$ to have positive and negative values. We denote n^+ the number of times the difference has a positive value and n^- the number of times the difference has a negative value so that $n = n^+ + n^-$. The test statistic k is the smallest of the two sums: $k = \min(n^+, n^-)$. The cumulative probability of the binomial distribution is defined as

$$f_c(k; n, p) = \sum_{i=0}^k \binom{n}{i} p^i (1-p)^{n-i}, p = 0.5$$

The two-tailed p-value equals $2 \times f_c(k; n, p)$. If the p-value is smaller than or equal to the significance level $\alpha = 0.001$, we reject the null hypothesis that the performance of the setup is different from the baseline due to chance only.

Normal approximation of the binomial distribution is used for larger samples $n \geq 25$ with the standard score for the Z statistic:

$$Z = \frac{k - \mu}{\sigma}, \mu = \frac{n}{2}, \sigma = \sqrt{\frac{n}{4}}$$

The p-value can be found in the table for the Z statistic.

Paired Wilcoxon signed-rank test While the sign test measures how often one setup outperforms the other, the Wilcoxon signed-rank test captures to what degree one model is better than the other. In the case of exact match and coverage as defined in our experiments, this aspect is not very interesting on its own as the differences are almost always $+1$ or -1 , but we use several tests to verify the results of one another. The null hypothesis H_0 is that the differences $a_i - b_i$ have a median value of zero. All the differences are assigned ranks according to their absolute values, $|a_i - b_i|$. If two or more differences are equal, the tied differences receive an average of the ranks they would be set if they were not equal. For example, if the lowest difference value

is equal to 1 and there are five instances of such a difference, then if the tied differences had different values they would be assigned ranks 1, 2, 3, 4 and 5, therefore an average rank that should be assigned is 3 ($\frac{1+2+3+4+5}{5} = 3$)

We denote W^+ as the sum of all the ranks associated with the positive differences ($a_i - b_i > 0$) and W^- as the sum of all the ranks associated with the negative differences ($a_i - b_i < 0$), and the test statistic W is the smaller of the two sums, $W = \min(W^+, W^-)$. Looking up the W statistic in the Wilcoxon distribution table as well as the size of the sample set n , we can determine the p-value and reject the null hypothesis if the p-value is smaller or equal to the significance level $\alpha = 0.001$.

For larger samples with $n \geq 20$ the binomial distribution is approximated with the normal distribution with the Z statistic computed by the following formula:

$$Z = \frac{k - \mu}{\sigma}, \mu = \frac{n(n+1)}{4}, \sigma = \sqrt{n(n+1)(2n+1)/24}$$

and the p-value is determined from the table for the Z statistic.

Paired t-test on two related samples Unlike the non-parametric signed and Wilcoxon signed-rank tests, in the paired t-test the differences $a_1 - b_1, \dots, a_m - b_m$ are assumed to fit the normal distribution. The null hypothesis H_0 is that the mean of differences $a_i - b_i$ has the value of zero. A statistic t with $n - 1$ degrees of freedom is defined by the formula

$$t = \frac{\bar{X}_D}{s_D / \sqrt{n}},$$

$$\bar{X}_D = \frac{\sum_{i=1}^n (a_i - b_i)}{n} = \bar{A} - \bar{B}$$

$$\hat{A}_i = a_i - \bar{A}, \hat{B}_i = b_i - \bar{B}$$

$$s_D = \sqrt{\frac{\sum_{i=1}^n ((a_i - b_i) - \bar{X}_D)^2}{n - 1}} = \sqrt{\frac{\sum_{i=1}^n (\hat{A}_i - \hat{B}_i)^2}{n - 1}},$$

$$t = (\bar{A} - \bar{B}) \sqrt{\frac{n(n-1)}{\sum_{i=1}^n (\hat{A}_i - \hat{B}_i)^2}}$$

where \bar{X}_D is the sample mean of differences, \bar{A} is the sample mean of a sequence A, \bar{B} is the sample mean of a sequence B, s_D is the sample standard deviation of differences, n is the sample size. Looking up the t-distribution table for the t statistic with $n - 1$ degrees of freedom will give the p-value. If the p-value is smaller than or equal to the significance level $\alpha = 0.001$, we reject the null hypothesis.

6.4 Experimental setup

An interesting research question of parser integration is how to choose high-quality dependencies in order to maximize the benefits of combination. As shown by Sagae et al. (2007), using the complete set of dependencies generated by the dependency parser as hard constraints

for HPSG parsing reduces coverage drastically. Their best result was achieved by running a parser ensemble of shift-reduce and graph-based parsers and choosing only those dependencies on which both agree.

In the present study we propose three different approaches as well as a combination of them to select the most reliable bilexical dependencies: filtering, confidence thresholding and ensemble. With the term *filtering* we define choosing only dependencies (i) spanning up to a maximum number of tokens; (ii) of a specific dependency type; (iii) with a dependent of a certain PoS type; or (iv) a combination of these criteria. We identify filtering parameters with a static analysis of dependencies on the development set in terms of precision and annotation rate without running the HPSG parser with these constraints. *Precision* is the proportion of correct system dependencies among all system dependencies and *annotation rate* is the proportion of input tokens that are assigned heads, e.g. the proportion of system dependencies among all tokens in the test set.

Confidence thresholding is an approach in which we rely on the functionality of some dependency parsers to produce per-dependency confidence scores and once again we tune a score threshold by performing a static analysis on the development set. In addition, we study the correlation of per-dependency scores with the correctness of attachment and labeling of individual dependencies.

Our *ensemble* method is similar to the method described in Sagae et al. (2007), as we use several parsers and choose dependencies by unweighted voting, but we explore a wider range of different parsing approaches and pay particular attention to parser efficiency.

In our experiments we use bilexical dependencies to restrict the search space of the HPSG parser PET. In this chapter we switch to the ERG version 1214 (which is a minor patch release of the ERG 1212 that was used in the previous chapters) and DeepBank 1.1 (a version following DeepBank 1.0 used in previous chapters). Since we started our experimentation before the final versions of the grammar and treebank were published, all our tuning experiments are performed on a pre-release versions of the ERG 1214 and DeepBank 1.1 using sections 0-19 for training and section 20 for development. The final testing is performed on the released versions of the ERG and DeepBank using sections 0-20 for training and 21 for testing. Out-of-domain experiments are carried out on the same parts of the Redwoods treebank as the out-of-domain testing in Chapter 5: an early essay on open source software “Cathedral and the Bazaar” by Eric Raymond (CB); a fragment of the SemCor corpus (SC); data from the VerbMobil corpus, a collection of transcribed spontaneous speech recorded in a dialogue task (VM); and part of the Wikipedia-derived WeScience Corpus (WS). For simplicity, we limit all the evaluations of the PET parser integrated with dependency constraints to the sets of fully disambiguated sentences with a unique gold standard analysis. In contrast to previous chapters, in this study we use only PTB-style tokenization. Table 6.1 shows the data sets used in the present study and their sentence counts.

In our experiments, bilexical dependencies are generated by an assortment of state-of-the-art dependency parsers employing quite different parsing algorithms: the transition-based Malt (Nivre et al., 2007b) version 1.7.2, the graph-based MST (McDonald et al., 2005b) version 0.5.0, the transition-based parser with a graph-based model of Bohnet and Nivre (2012) (dubbed B&N; transition-1.30.jar, beam 80), the transition-based Mate (Bohnet, 2010) (anna-3.61.jar, beam 80) and the linear programming Turbo (Martins et al., 2013) version 2.2.0. Malt,

Dataset	# sentences	# tokens	Average sentence length
DeepBank pre-released			
WSJ 0-19	34,971	781,807	22.4
WSJ 20	1,782 (1,778)	39,963	22.4
DeepBank released			
WSJ 0-20	37,351	846,951	22.7
WSJ 21	1,476 (1,399)	33,570	22.7
Redwoods			
CB	596 (596)	12,532	21.0
SC	861 (860)	15,426	17.9
VM	986 (985)	8,728	8.9
WS	449 (446)	8,325	18.5

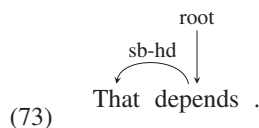
Table 6.1: Sentence and token counts and average sentence length for data sets of DeepBank 1.1 and Redwoods prepared with ERG 1214: pre-released version of DeepBank is employed for tuning, released version of DeepBank is applied for testing and Redwoods is used for out-of-domain experiments. For the sets on which we test parser integration setups we show in the brackets the number of fully disambiguated sentences with a unique gold standard analysis

MST, Mate and Turbo have been run with PTB tags produced by the TnT part-of-speech tagger (Brants, 2000), while B&N predicts PoS tags during parsing. In all cases, the tokenization has been performed with the Regular Expression Pre-Processor (REPP) of Drìdan and Oepen (2012).

Parser integration is realized via the so-called YY input mode¹ of the PET parser and all experimentation is based on PTB-style tokenization. Using ERG-style tokenization as PET input would not work, because prefix and suffix punctuation would cause the named entity recognition to fail, for example the date “(01.02.2000)” would not be identified as a date because the regular expression pattern does not include brackets. The YY format allows us to supply PET not only with input tokens, but also with PoS tags (and their probabilities) and labeled dependencies. Example (72) shows a representation of the gold dependency analysis in the YY input mode of the sentence “That depends.” from the VM part of the Redwoods treebank (with sentence identifier, dependencies and tokens shown on separate lines for clarity) and Example (73) illustrates the graphical representation of the corresponding dependencies.

(72) [1320384] |
 (0, 0, 1, 1, "[→ |sb-hd|5]", 0, "null")
 (1, 1, 2, <0:4>, 1, "That", 0, "null", "WDT" 0.8254 "IN" 0.1448 "DT" 0.0298)
 (2, 2, 3, 1, "[→ |root]", 0, "null")
 (3, 3, 4, <5:12>, 1, "depends", 0, "null", "VBZ" 1.0)
 (4, 4, 5, <12:13>, 1, ".", 0, "null", "." 1.0)

¹<http://moin.delph-in.net/PetInput> Accessed: 14 August 2015.



Number “1320384” is a sentence identifier in the treebank. In the present sentence there are three “regular” input tokens (“That”, “depends”, “.”) and two dependency “pseudo”-tokens (“sb-hd”, “root”) which are depicted in round brackets with dependency annotations preceding dependent tokens. The general YY format is the following:

(id, start, end, [link,] path+, surface, ipos, lrule+[, pos p+])

The first field in round brackets, *id*, is an identifier, e.g. “0” for the dependency token “sb-hd”, “3” for the token “depends”. The second and third fields are *start* and *end* vertices, e.g. “0” and “1” for the dependency “sb-hd”, “3” and “4” for the token “depends”. The field *link* is only used for input tokens and stands for the character-based span of the token, e.g. “<5:12>” for the token “depends” which corresponds to the sub-string from position “5” to “12”. The field *path* stands for membership in one or more paths through a word lattice, e.g. “1” for all tokens and dependencies in our example. For regular tokens the *surface* field provides the original string. For dependencies the *surface* field records directionality (e.g. → - incoming dependency), dependency label (e.g. “sb-hd”) and a start position in the character-based identifier of the head for non-root dependencies (e.g. “5” for dependency “sb-hd”, which denotes that the character-based identifier of the head token starts from position “5”, and from this information we can derive that the head word for this dependency is “depends” because its character-based identifier starts from position “5”: “<5:12>”). The *ipos* field can be used to provide information about morphological segmentation and it denotes the position to which orthographemic rules apply, but if we choose to rely on standard orthographemic annotation provided by the grammar, this field should always be set to “0”. The *lrule* field can be employed to specify lexical rules for morphological analysis, but to activate standard morphological analysis of PET, the value should be set to “null”. The *pos* field is used only for tokens to specify the sequence of assigned PoS tags with their corresponding probabilities, e.g. the token “That” is assigned three PoS tags: “WDT” with probability 0.8254, “IN” with probability 0.1448, and “DT” with probability 0.0298.

It is important to note that we ignore dependencies that are not “native” to the ERG derivation trees, but are created artificially to comply with PTB-style tokenization (see Chapter 3). There are three types of such dependencies: “PUNCT” (punctuation), “MWE” (multiword expression) and “NEG” (negation). In the example above, there would be a dependency with the head “depends”, dependent “.” and label “PUNCT” in the gold DT tree, but it is not included in the YY input mode, because parser-internally the two PTB tokens are treated as a single ERG token, where the period functions much like a suffix.

In addition, we exclude incoming dependencies to words containing hyphens (e.g. “couch-potato”) and slashes (e.g. “Cities/ABC”) due to complications in tokenization. Table 6.2 that we have already seen in Chapter 3, explains the pipeline of the tokenization process in PET starting from initial to lexical tokens. Initial tokens are produced automatically with REPP and emulate PTB-style tokenization. We extract these tokens from the gold treebanks and give them as input to the dependency parsers and then to PET in YY input mode. Lexical tokens are

Raw	‘Sun-filled’, well-kept Mountain View. REPP
initial tokens	[‘], [Sun-filled], [’], [,], [well-kept], [Mountain], [View], [.] chart-mapping
internal tokens	[‘Sun-], [filled’], [well-], [kept], [Mountain], [View.] lexicon lookup
lexical tokens	[‘sun-], [filled’], [well- kept], [Mountain View.], [well-], [kept], [Mountain], [View.]

Table 6.2: Tokenization pipeline during parsing with PET (Dridan, 2013b)

ERG derivation tree leaves corresponding to the lexicon entries that comply with the ERG-style tokenization. For example, the word “couch-potato” is represented as one initial token (“couch-potato”) but two lexical tokens (“couch-”, “potato”) and the word “Cities/ABC” is represented as one initial token (“Cities/ABC”) but three lexical tokens (“Cities”, “/”, “ABC”) in the ERG analysis. However, it is not a rule that all words with a hyphen or slash are split into separate lexical entries, as the representation of the word in the ERG-style tokenization depends on how it is recorded in the lexicon of the grammar. When working with outputs from dependency parsers, we do not know how the initial tokens in the parser output would correspond to the lexical ones in the derivation tree in the PET analysis, and for this reason it is safer to discard incoming dependencies on such tokens in the parser integration setup.

The mechanism of introducing dependency constraints is realized via a generic, declarative interface developed by Dan Flickinger and Stephan Open. The interface is incorporated in the ERG grammar 1214 and is compatible with such ERG parsers as PET and ACE (Packard, 2011). The possibility of using this interface was one of the reasons to switch to ERG version 1214 in this chapter. In the following, we describe the details of the internal realization of parser integration.

PET has a standard mechanism called Token Mapping (Adolphs et al., 2008) for adaptation of input tokens delivered by external preprocessors to the expectations of the grammar. Token mapping is one of the phases of the chart mapping process which is a mechanism for the non-monotonic, rule-based manipulation of chart items that are described by feature structures². Input tokens are formalized in token feature structures and token manipulation is realized by formal devices of the grammar such as unification and co-indexation. During the phase of Token Mapping PET operates on internal tokens that conform ERG derivation tree leaves. The bilexical dependency constraints are enforced as features on the internal tokens by regular unification and are used to instantiate lexical tokens during the lexicon lookup. Lexical tokens are annotated with information about their identifier, expected head (target) and dependency type (label). All syntactic rules percolate these annotations in a special feature called “-DT” from the head to the parent node and require the following unifications: (a) the target value on all non-head daughters with the token identifier value on the head daughter; (b) the construction type (i.e. the identity of the rule) with the label value on all non-head daughters. As a result

²http://moin.delph-in.net/Chart_Mapping Accessed: 14 August 2015.

	EM	C	T_P
baseline	43.76	94.7	11.62
upper bound	80.88	100	0.85

Table 6.3: Baseline and upper bound for parser combination on the development set. “EM” is the exact match, “C” is the coverage, T_P is the running time of PET

the chart will only contain instances of syntactic rules that are compatible with the dependency annotations. All other items fail in unification, i.e. are in fact never built, rather than pruned.

In the following experiments, the baseline for the parser integration setup is parsing with PET without restricting the search space with dependencies and the upper bound is parsing with gold dependencies.

6.5 Tuning

With the goal of reducing the large space of candidate combination methods and selecting only the most promising setups, we perform tuning on the development set which is the section 20 of DeepBank (the test set is not used in the tuning experiments). This round of experiments is performed with the pre-released versions of the grammar and treebank—the ERG 1214 and DeepBank 1.1 correspondingly.

Table 6.3 shows the baseline (standard parsing with PET) and the upper bound (restricting the search space of the PET parser with gold dependencies) on the development set. The maximum time that PET can use to process a sentence is a user-defined parameter and for the baseline it is generously set to 5 minutes, the standard development configuration. It might seem surprising that the baseline setup does not have a complete coverage since it is built without pruning similarly to the gold-standard, except that the best analysis is chosen automatically rather than manually. However the gold treebank is created on the faster machines with even more generous recourse limits.

We experiment with several approaches to preparing bilexical dependencies for limiting the search space of the HPSG parser: a) generating dependencies with an individual data-driven dependency parser and then implementing either filtering parameters or confidence thresholding to choose only the high-confidence dependencies; b) producing dependencies with ensembles of parsers. For the integration of PET with an individual dependency parser we have chosen the Turbo, B&N and MST parsers, as the first one is the most efficient state-of-the-art syntactic analyzer, and the latter two provide per-dependency probability scores that we employ for confidence thresholding. For ensemble integration with PET, the following groups of parsers are used:

BMT B&N, Mate, Turbo;

MMMT Malt, MST, Mate, Turbo;

BMMMT B&N, Malt, MST, Mate, Turbo.

	LAS	UAS	LACC	Time, self-eval.	Time, average
Malt	89.00	91.48	91.16	12.141	0.0068
MST	90.62	93.23	92.60	295.585	0.1859
B&N	92.15	94.29	93.81	2501.4	1.4037
Mate	92.07	94.12	93.81	380.8	0.2137
Turbo	91.72	94.02	93.52	171.43	0.0962

Table 6.4: Evaluation of individual parsers including punctuation on the development set (section 20 of DeepBank). Self-evaluation time is measured in seconds, average time per sentence is measured in seconds per sentence

The choice of ensemble groups is motivated by the following factors: the first ensemble includes the newer state-of-the-art statistical tools therefore the expectation is that it will produce more precise dependencies, the second ensemble excludes the slowest parser B&N and for this reason we anticipate greater efficiency gains, the third ensemble includes all five parsers and we hypothesize that only high-confidence dependencies will be selected with a strict voting approach.

The individual performance of the parsers on the development set of DeepBank is presented in Table 6.4. All the parsers provide self-evaluation of how much total time it takes to parse a given input set (see the column “Time, self-eval.”). Since we are more interested in the average parsing time per sentence, we divide the self-evaluation timing on the total size of the development set which constitutes 1782 sentences (see the column “Time, average”). As the baseline time of parsing with PET is 11.62 seconds per sentence (see Table 6.3) and the slowest dependency parser among the five, B&N, has the time of 1.4037 seconds per sentence, running the members of an ensemble in parallel we can expect significant speed up for the integrated setup, even for ensembles including the slowest data-driven parser. However, the baseline measurement is not directly comparable to the time estimates of the individual parsers because of differences in evaluation approaches and the clusters on which the computation is performed. All time evaluations for the dependency-based parsers are performed on a single CPU of the Titan computer cluster of the University of Oslo and the parsers output a self-estimation of how much time is elapsed for the parsing of a given input set. Efficiency estimations for PET (including baseline and upper bound) are computed over six concurrent instances of the parser, each parsing different sentences, on a computer node of the Abel computer cluster of the University of Oslo and PET measures time for each sentence and the final evaluation is an average per sentence. We argue that the average evaluation per sentence realized in PET is a more practical predictor for parse times of new and variable-sized data sets than the total parsing time of a given data set because the former is more general as it abstracts from a particular treebank.

The tuning is organized in two phases: firstly, for all the dependency setups we carry out a static analysis using precision and annotation rate as evaluation measures and choose the best configurations; secondly we integrate the selected setups with PET and single out integration methods for in-domain testing based on coverage, efficiency and exact match.

	EM	C	T_P
baseline	43.76	94.7	11.62
Turbo	33.52	65.4	0.82
B&N	37.18	71.0	0.84
BMT, 3 votes	43.59	89.3	1.07
MMMT, 4 votes	41.73	90.7	1.19
BMMMT, 3+ votes	38.25	74.2	0.90
BMMMT, 5 votes	43.36	93.4	1.29

Table 6.5: Results of parser combination without filtering on the development set. Setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and running time of PET (“ T_P ”). Number of votes specifies how many parsers in the ensemble should agree on a labeled dependency.

6.5.1 Filtering parameters

Table 6.5 reports the performance of different parsing combination approaches that we will discuss in detail in the following sections. Some configurations already deliver quite good results, for example the BMMMT ensemble with a requirement of all parsers in the ensemble to agree on a dependency. We observe a great speed-up of parsing but at a cost of reduced coverage. The motivation to define and tune the filtering parameters is preserving or even improving coverage while still guaranteeing reasonable efficiency gains over the baseline. The filtering parameters are defined and continuously specified during both phases of the tuning process based on an error analysis of the individual parsers, ensembles and integration with PET.

Filtering by dependency length. The first filtering criterion we consider is the maximum dependency length, e.g. the maximum number of tokens that a given dependency spans over. As it has been shown in previous studies (Eisner and Smith, 2010), long dependencies are empirically rare and therefore statistical parsers are more prone to make mistakes on them. We experiment with the length constraints of 10, 5 and 3 during tuning. Table 6.6 shows pairs of setups where we vary only the dependency length parameter (indicated by L). With tighter upper bounds on dependency length we filter dependencies more aggressively which results in lower annotation rate (i.e. we select fewer dependencies), but it has positive effects on the coverage of the integrated configuration as we can see from Table 6.6, where the best results are obtained for dependency length 3.

Filtering by part-of-speech tag. Some constructions, such as coordination and PP-attachment, are notoriously difficult for parsers and consequently by filtering by part-of-speech we hope to identify more accurate dependencies. We select several PTB lexical types that guarantee minimum 90% accuracy over CPOSTAGs for individual parsers on the development set: *VB* (verb, base form), *NNP* (proper noun, singular), *PRP* (personal pronoun), *PRP\$* (possessive pronoun), *DT* (determiner), *WDT* (wh-determiner), *CD* (cardinal number), *\$* (dollar). In the

	P	AR	EM	C	T_P
BMT, 3 votes, L10	96.21	75.95	43.25	89.8	1.10
BMT, 3 votes, L5	96.48	71.95	42.63	90.9	1.25
MMMT, 4 votes, L10	96.56	72.31	41.56	90.8	1.24
MMMT, 4 votes, L5	96.71	69.14	41.11	91.5	1.37
BMMMT, 5 votes, L10	97.24	70.86	43.14	93.5	1.32
BMMMT, 5 votes, L5	97.35	67.81	42.80	93.9	1.46
BMT, 3 votes, prec80, L5	97.29	62.81	42.41	92.0	1.68
BMT, 3 votes, prec80, L3	97.38	55.15	42.91	93.1	2.04
MMMT, 4 votes, prec80, L5	97.41	61.18	42.07	92.7	1.81
MMMT, 4 votes, prec80, L3	97.46	53.96	42.41	93.4	2.17
BMMMT, 5 votes, prec80, L5	97.88	60.21	43.42	94.4	1.89
BMMMT, 5 votes, prec80, L3	97.91	53.14	43.59	94.9	2.28

Table 6.6: Tuning filtering parameter length of dependency. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “L3”, “L5”, “L10” means that dependencies spanning more than 3, 5 or 10 tokens correspondingly are filtered out.

setup where we combine the Turbo parser with PET, we add two extra PoS tags *VBN* and *VBZ* because the accuracy of Turbo on these two PoS is also above the threshold of 90%. For the ensemble BMMMT, this criterion is too strict as it filters out all the dependencies. Table 6.7 demonstrates the effect of this filtering criterion: by activating the filtering parameter by PoS we select significantly fewer dependencies which results in low annotation rate in the static analysis and slower parsing in the integrated setup, however we observe that the selected dependencies are more accurate which is indicated by precision in the static analysis and we detect notable gains for the coverage of the integrated setup.

Filtering by dependency type. As shown by McDonald and Nivre (2007), properties of dependency types can have substantial impact on errors in data-driven dependency parsing. For pruning by dependency type we attempt several strategies chosen from the static error analysis of individual dependency parsers on the development set:

- filtering dependencies that have a frequency less than 300 and precision less than 70% in the error analysis of the individual parsers that constitute the BMMMT ensemble; Table 6.8 shows that this filtering approach does not give notable advantages neither in the precision of selected dependencies nor in the coverage nor in the exact match of the parser integration.
- selecting only dependency types for which the precision is minimum 80% in the error analysis of the individual parsers that constitute the BMMMT ensemble; Table 6.9

	P	AR	EM	C	T_P
BMT, 3 votes	96.12	77.62	43.59	89.3	1.07
BMT, 3 votes, pos8	97.82	27.31	43.31	97.6	5.40
MMMT, 4 votes	96.52	73.51	41.73	90.7	1.19
MMMT, 4 votes, pos8	97.84	26.72	43.25	97.6	5.79
Turbo, prec80	93.51	71.96	36.22	73.1	1.25
Turbo, prec80, pos10	96.00	30.85	42.01	93.8	4.12

Table 6.7: Filtering parameter part-of-speech tag of dependent. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “pos8”, “pos10” means that only dependencies for dependent token of one of the 8 or 10 PoS are selected; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble

demonstrates that with this criterion we select more accurate dependencies and improve the coverage of the integrated setup.

- selecting only the 11 most accurate (with precision higher than 98%) dependency types from the error analysis of the BMMMT ensemble: N-NUM, ROOT, SB-HD, HD-HD, J-N, SP-HD, FLR-HD, N-J, NUM-N, PP-PP, VP-VP; As follows from Table 6.10 this filtering method improves precision in the static analysis and coverage of the integrated setup.
- selecting only the 10 most frequent and accurate (with precision higher than 93%) dependency types from the error analysis of the BMT ensemble: N-NUM, ROOT, SB-HD, HD-CMP, SP-HD, NP-HDN, N-HDN, MRK-NH, NUM-N, HD-PCT; From Table 6.11 we find out that this filtering approach is not very effective for our purposes.
- selecting only the 8 most accurate (with precision higher than 90%) and frequent dependency types from the error analysis of individual parsers included in the BMMMT ensemble: N-NUM, SB-HD, HD-CMP, SP-HD, NP-HDN, N-HDN, NUM-N, HD-PCT; From Table 6.12 we do not have enough evidence that this filtering parameter is useful.
- selecting only the 7 most accurate (with precision higher than 98%) dependency types from the error analysis of the MST parser with the KD-Fix algorithm (Mejer and Cramer, 2010, 2012): N-NUM, ROOT, SB-HD, HD-HD, J-N, SP-HD, HDN-NP. With this filtering method we try to maximize coverage for the settings of the MST parser with confidence thresholding.

PET integration with Turbo We have chosen the Turbo parser for direct combination with PET because it shows state-of-the-art accuracy and efficiency. Table 6.14 displays the experimentation with filtering the Turbo output and its integration with PET. Based on the tuning

	P	AR	EM	C	T_P
BMT, 3 votes	96.12	77.62	43.59	89.3	1.07
BMT, 3 votes, freq300, prec70	96.60	72.85	42.91	90.3	1.22
MMMT, 4 votes	96.52	73.51	41.73	90.7	1.19
MMMT, 4 votes, freq300, prec70	96.90	69.66	42.29	91.5	1.36
BMMMT, 5 votes	97.20	72.01	43.36	93.4	1.29
BMMMT, 5 votes, freq300, prec70	97.50	68.36	43.64	93.8	1.48

Table 6.8: Filtering dependencies that have frequency less than 300 and precision less than 70%. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “freq300” means that only dependencies with frequency more than 300 are selected; “prec70” means that only those dependency types were selected for which precision is 70% or higher on the development set for all the parsers in the ensemble

results, we have chosen the configuration of the Turbo parser with filtering parameters involving ten most accurate dependency types, ten PoS tags and maximum length of dependency of five tokens for further experimentation on the test set as this setup shows the best values of precision of dependencies, coverage and exact match on the development set.

6.5.2 Confidence thresholding

We define confidence thresholding as an approach to select high-quality bilexical dependencies by setting a threshold on the per-dependency probability scores produced by a parser. We explore this method for B&N and MST since these parsers estimate confidence for each dependency.

Confidence thresholding with B&N

The B&N parser can generate per-dependency scores. In the current subsection we attempt to use these scores to detect which dependency types the B&N parser predicts most reliably.

The parser produces three per-dependency probabilities: e-, x- and a-scores. The **e-score** is the score of the completion model - a graph-based score. The **x-score** is the edge-based score coming from the transition-based features. The **a-score** is an accumulated transition-based score which includes shifts, swaps and the edge score.

We evaluate the correlation of the per-dependency e-, x- and a-scores with the correctness of the attachment and dependency labels using the Pearson correlation coefficient which measures the linear relationship between two datasets. This statistical measure ranges between -1 and 1 with 0 implying no correlation, 1 implying perfect positive correlation (as the variables in one dataset increase, the variables in the other dataset also increase and vice versa) and -1 implying perfect negative correlation (as the variables in one dataset increase the variables in the other dataset decrease and vice versa). In order to compute the Pearson correlation, we collect six lists

	P	AR	EM	C	T _P
BMT, 3 votes, L5	96.48	71.95	42.63	90.9	1.25
BMT, 3 votes, L5, prec80	97.29	62.81	42.41	92.0	1.68
MMMT, 4 votes, L5	96.71	69.14	41.11	91.5	1.37
MMMT, 4 votes, L5, prec80	97.41	61.18	42.07	92.7	1.81
BMMMT, 5 votes, L5	97.35	67.81	42.80	93.9	1.46
BMMMT, 5 votes, L5, prec80	97.88	60.21	43.42	94.4	1.89

Table 6.9: Selecting only dependency types for which precision is minimum 80%. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“T_P”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “L5” means that dependencies spanning more than 5 tokens are filtered out

	P	AR	EM	C	T _P
BMMMT, 5 votes	97.20	72.01	43.36	93.4	1.29
BMMMT, 5 votes, dep11	98.24	24.91	44.26	98.1	4.86

Table 6.10: Selecting only dependencies of 11 types. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“T_P”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “dep11” means that only dependencies of one of the 11 dependency types are selected

	P	AR	EM	C	T _P
BMT, 3 votes	96.12	77.62	43.59	89.3	1.07
BMT, 3 votes, dep10	96.71	59.40	41.17	89.9	1.68
BMT, 3 votes, L5	96.48	71.95	42.63	90.9	1.25
BMT, 3 votes, L5, dep10	96.84	56.58	41.28	90.8	1.87
MMMT, 4 votes	96.52	73.51	41.73	90.7	1.19
MMMT, 4 votes, dep10	97.03	57.37	41.45	91.6	1.80
MMMT, 4 votes, L5	96.71	69.14	41.11	91.5	1.37
MMMT, 4 votes, L5, dep10	97.09	55.06	41.34	92.0	2.00
BMMMT, 5 votes	97.20	72.01	43.36	93.4	1.29
BMMMT, 5 votes, dep10	97.49	56.46	42.41	93.0	1.93
BMMMT, 5 votes, L5	97.35	67.81	42.80	93.9	1.46
BMMMT, 5 votes, L5, dep10	97.53	54.22	42.29	93.4	2.13

Table 6.11: Selecting only dependencies of 10 types. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“T_P”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “L5” means that dependencies spanning more than 5 tokens are filtered out; “dep10” means that only dependencies of one of the 10 dependency types are selected

	P	AR	EM	C	T_P
BMT, 3 votes	96.12	77.62	43.59	89.3	1.07
BMT, 3 votes, dep8	96.70	53.12	41.45	91.1	1.94
MMMT, 4 votes	96.52	73.51	41.73	90.7	1.19
MMMT, 4 votes, dep8	97.03	51.50	41.90	92.6	2.11
BMMMT, 5 votes	97.20	72.01	43.36	93.4	1.29
BMMMT, 5 votes, dep8	97.47	50.72	42.69	94.0	2.22

Table 6.12: Selecting only dependencies of 8 types. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “dep8” means that only dependencies of one of the 8 dependency types are selected

	P	AR	EM	C	T_P
MST, KD-Fix*0.05*50, thr1	96.65	57.16	38.19	89.2	2.44
MST, KD-Fix*0.05*50, thr1, dep7	98.67	18.88	43.59	96.9	6.46
MST, KD-Fix*0.05*100, thr1	96.92	53.04	38.92	90.5	2.79
MST, KD-Fix*0.05*100, thr1, dep7	98.78	17.84	43.53	96.7	6.07

Table 6.13: Selecting only dependencies of 7 types. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). KD-Fix*0.05*50 stands for the KDF-Fix algorithm with the parameters standard deviation of 0.05 and K of 50; “thr” stands for the threshold on per-dependency probability score; “dep7” means that only dependencies of one of the 7 dependency types are selected

	P	AR	EM	C	T _P
baseline	0	0	43.76	94.7	11.62
Turbo	90.87	86.39	33.52	65.4	0.82
Turbo, L5	92.13	78.60	33.80	70.1	0.99
Turbo, dep10	93.23	63.86	36.28	74.2	1.33
Turbo, dep10, L5	93.86	60.23	36.78	77.4	1.54
Turbo, pos10	95.01	33.29	40.78	91.8	3.44
Turbo, pos10, L5	95.57	31.79	41.28	93.4	4.00
Turbo, dep10, pos10	96.00	30.13	42.07	93.9	4.23
<i>Turbo, dep10, pos10, L5</i>	96.34	29.10	42.18	94.9	4.57
Turbo, dep10 or pos10	93.61	62.92	35.21	72.4	1.27
Turbo, dep10 or pos10, L5	92.90	67.02	36.00	76.0	1.45
Turbo, prec80	93.51	71.96	36.22	73.1	1.25
Turbo, prec80, L5	94.22	66.97	36.90	76.6	1.41

Table 6.14: Parser integration: Turbo and PET. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“T_P”). “dep10” means that only dependencies of one of the 10 dependency types are selected; “pos10” means that only dependencies for dependent token of one of the 10 PoS are selected; “L5” means that dependencies spanning more than 5 tokens are filtered out; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble

	e-scores	x-scores	a-scores
correctness of attachment and labeling	0.1878	0.0651	0.1088
correctness of attachment	0.1551	0.0514	0.1180
correctness of labeling	0.1827	0.0603	0.0857

Table 6.15: Correlation of per-dependency confidence scores of the B&N parser with correctness of attachment and labeling

representing the correctness of attachment, labeling and both attachment and labeling prediction (with the value 1 for correct predictions and the value 0 for incorrect predictions) and the e-, x- and a-score values for each token in the development set. We further compute the Pearson correlation coefficients for each of the three former lists with respect to each of the three later lists, using the formula:

$$\frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

where n is the number of tokens in the development set, x_i are the values of one list and y_i are the values of the other list. The results shown in Table 6.15 suggest that there is no strong correlation between probability scores and correctness of dependencies (since correlation scores are higher than 0 but lower than 0.5). We decided nevertheless to tune the thresholds for the scores in order to extract at least a small number of high-quality dependencies.

Figure 6.1 illustrates the effect of optimizing the B&N probability scores for precision on the annotation rate. We observe an unsurprising reverse relationship between the two metrics which shows that the more accurate dependencies we intend to select from the output of B&N, the lower overall annotation rate is. We seek to find thresholds for the probability scores that assure certain levels of precision and annotation rate. Table 6.16 summarizes the results of our study. It is slightly confusing, but our interpretation of the annotation rate measure (which we will refer as “PD”—the proportion of selected dependencies with respect to all dependencies generated with B&N) in this context differs from the evaluation metric annotation rate (“AR”) that we use to evaluate the resulted setups. In these early experiments we included dependencies with labels “PUNCT”, “MWE”, “NEG” and incoming dependencies to words with hyphens and dashes for the calculation of the proportion of selected dependencies (“PD”) for thresholding the probability scores of B&N. But for the evaluation of the setups the metric annotation rate (“AR”) is computed after filtering such dependencies. For this reason in Table 6.16 we observe that when we set the level of “PD” to 20% (“PD20”) to threshold e-, x- and a-scores, the evaluation results in an “AR” of only 7.58%.

As one can eyeball from Table 6.16 we achieve a speed-up over the baseline in all the tested setups, however the coverage is at baseline level or higher only when less than 28% of the dependencies are selected for integration with PET. Even though our initial analysis of the reliability of per-dependency probability scores was not encouraging, there are two interesting setups that we have chosen for experiments on the test set: (1) B&N with a filtering parameter selecting only dependency types with minimum precision of 95% and “PR40” which provided the best efficiency with coverage on the baseline level; and (2) B&N with a filtering parameter

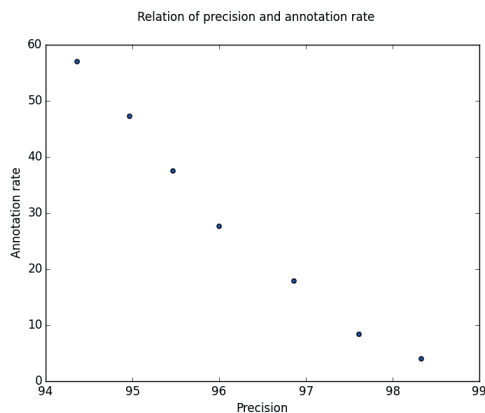


Figure 6.1: Precision versus annotation rate for B&N on the development set

	P	AR	EM	C	T_P
baseline	0	0	43.76	94.7	11.62
B&N, prec95, PD20	97.61	8.36	43.36	95.7	7.58
B&N, prec95, PD40	96.00	27.65	43.31	94.7	3.42
B&N, prec95, PD60	94.97	47.20	42.01	88.5	1.69

Table 6.16: Confidence thresholding with B&N. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). “prec95”, “PD20” means that the values of e-, x- and a-scores are tuned so that at least 20% of all the generated dependencies are selected for integration with PET and the accuracy of the selected dependencies is at least 95%

selecting only dependencies with minimum precision of 95% and “PR20” which offered the best coverage in our tuning experiments.

Confidence thresholding with MST

The MST parser can produce per-edge confidence scores indicating the parser’s confidence in the correctness of each predicted edge using the *K Draws by Fixed Standard Deviation (KD-Fix)* method (Mejer and Crammer, 2010). The confidence score of each edge predicted by the model is defined to be the fraction of parse trees containing this edge among the K trees (Mejer and Crammer, 2012):

$$v = \frac{j}{K}$$

where j is the number of parse trees that contain this edge ($j \in 0 \dots K$) so $v \in [0, 1]$. We set $K = 50$ following Mejer and Crammer (2012).

Similarly to the experiments on confidence thresholding with B&N, we compute the Pearson correlation for the MST per-dependency confidence scores and the correctness of the attachment

	confidence scores
correctness of attachment and labeling	0.4728
correctness of attachment	0.5216
correctness of labeling	0.3823

Table 6.17: Correlation of MST per-dependency confidence scores estimated with KD-Fix algorithm (standard deviation 0.05 and K 50) with correctness of attachment and labeling

and labeling. In accordance with Table 6.17 there is a moderate correlation between the MST confidence scores and the correctness of the attachment with the correlation value of 0.5216 (the Pearson correlation coefficient with a magnitude higher than 0.5 indicates that the two datasets can be considered moderately correlated).

In the instructions included in the MST parser package example values of the KD-Fix parameters are 0.05 for the standard deviation and 50 for K. While tuning the parameters for the KD-Fix algorithm we varied the standard deviation in the interval from 0 to 1 with the step of 0.1, and for each fixed standard deviation we tried K from 1 to 100 with the step of 1. The graphs in Figure 6.2 suggest that the value 0.05 for the standard deviation is optimal and for the parameter K value 50 is good while higher values can potentially improve the correlation, because of this reason we also experimented with K 100 in the subsequent experiments.

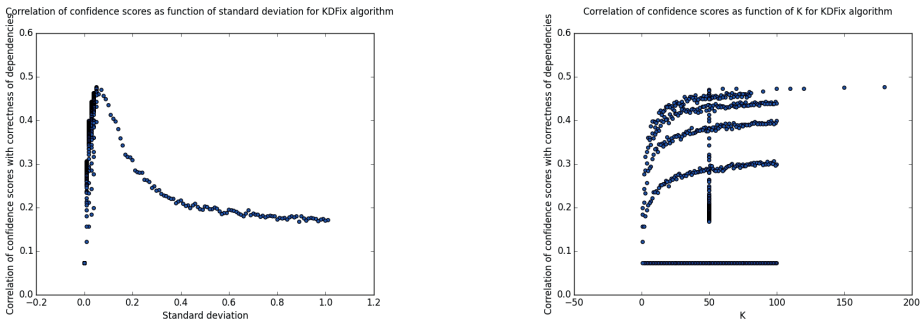


Figure 6.2: Tuning standard deviation and K of the KD-Fix algorithm on the development set. The graphs show the correlation of confidence scores with correctness of dependencies depending on different values of parameters

In the tuning phase, we have experimented with 0.98, 0.99 and 1 as the threshold for the per-dependency probability score, because we are only interested in high-confidence dependencies. We observed slight (however, maybe insignificant) improvements of precision in the static analysis when pruning dependencies based on a threshold of 1 rather than 0.98, and therefore decided to select only dependencies with probability 1.

Table 6.18 provides an overview of the tuning experiments on confidence thresholding of the MST parser and its integration with PET. For the experiments on the test set we have chosen three setups: (1) MST with standard deviation 0.05, K 50 and threshold probability 1, - maximizing coverage and exact match, (2) MST with standard deviation 0.05, K 50, threshold

	P	AR	EM	C	T_P
baseline	0	0	43.76	94.7	11.62
MST, KD-Fix*0.05*50, thr0.98	96.29	63.13	37.35	86.2	1.99
MST, KD-Fix*0.05*50, thr0.99	96.65	57.16	38.19	89.2	2.43
MST, KD-Fix*0.05*50, thr1	96.65	57.16	38.19	89.2	2.44
MST, KD-Fix*0.05*50, thr1, dep7	98.67	18.88	43.59	96.9	6.46
MST, KD-Fix*0.05*50, thr1, prec80, L5	97.58	48.00	40.89	91.8	3.51
MST, KD-Fix*0.05*50, thr1, pos8	98.14	22.37	44.42	97.2	6.11
MST, KD-Fix*0.05*50, thr1, L5	96.70	53.18	38.36	90.2	2.81
MST, KD-Fix*0.05*100, thr1	96.92	53.04	38.92	90.5	2.79
MST, KD-Fix*0.05*100, thr1, dep7	98.78	17.84	43.53	96.7	6.07
MST, KD-Fix*0.05*100, thr1, prec80, L5	97.67	45.14	41.06	93.1	3.65
MST, KD-Fix*0.05*100, thr1, pos8	98.34	21.29	43.03	96.7	6.35

Table 6.18: Confidence thresholding with MST. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). KD-Fix*0.05*50 stands for the KDF-Fix algorithm with the parameters standard deviation of 0.05 and K of 50, “thr” stands for the threshold on per-dependency probability score; “dep7” means that only dependencies of one of the 7 dependency types are selected; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

on probability 1, minimum precision of dependencies 80% from static analysis and maximum dependency length 5, - optimizing the parsing time of PET; (3) MST with standard deviation 0.05, K 100, threshold probability 1, and 7 dependency types chosen from the static analysis - offering very high precision of dependencies and compromised coverage, time and exact match.

6.5.3 Ensembles of parsers

For the ensemble method we have chosen five state-of-the-art data-driven dependency parsers: Malt, MST, B&N, Mate and Turbo, and experimented with three ensembles: 1) BMT (B&N, Mate, Turbo) as these three parsers are the most recent ones; 2) MMMT (Malt, MST, Mate, Turbo) as this ensemble excludes the slowest parser B&N; 3) BMMMT (B&N, Malt, MST, Mate, Turbo) joining the strengths of all five tools.

Table 6.19 sums up all configurations with the BMT ensemble and shows the results obtained during tuning on the development set. Based on these results, we have chosen three configurations for the experiments on the test set: BMT with agreement between all three parsers which is the fastest setup; BMT with 3 votes and 8 PoS which provides the best coverage; BMT with 3 votes, dependency type with minimum precision 80% from the static analysis and maximum length 5 which balances coverage and speed.

The tuning experiments with the ensemble MMMT are introduced in Table 6.20. For the

	P	AR	EM	C	T_P
baseline	0	0	43.76	94.7	11.62
<i>BMT, 3 votes</i>	96.12	77.62	43.59	89.3	1.07
BMT, 3 votes, L10	96.21	75.95	43.25	89.8	1.10
BMT, 3 votes, L5	96.48	71.95	42.63	90.9	1.25
<i>BMT, 3 votes, pos8</i>	97.82	27.31	43.31	97.6	5.40
BMT, 3 votes, pos8, L5	97.93	26.56	43.31	97.2	5.87
BMT, 3 votes, dep10	96.71	59.4	41.17	89.9	1.68
BMT, 3 votes, dep8	96.70	53.12	41.45	91.1	1.94
BMT, 3 votes, freq300, prec70	96.60	72.85	42.91	90.3	1.22
BMT, 3 votes, freq300, prec70, L5	96.77	68.59	42.80	91.5	1.37
BMT, 3 votes, dep10, L5	96.84	56.58	41.28	90.8	1.87
BMT, 3 votes, prec70, L5	96.77	68.59	42.80	91.5	1.37
<i>BMT, 3 votes, prec80, L5</i>	97.29	62.81	42.41	92.0	1.68
BMT, 3 votes, prec80, L3	97.38	55.15	42.91	93.1	2.04
BMT, 3 votes, dep10 or L5	96.41	74.77	42.69	90.2	1.17
BMT, 3 votes, prec70 or pos8, L5	96.75	68.89	42.74	91.3	1.35

Table 6.19: Ensemble BMT, tuning. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “dep10” means that only dependencies of one of the 10 dependency types are selected; “pos10” means that only dependencies for dependent token of one of the 10 PoS are selected; “L5” means that dependencies spanning more than 5 tokens are filtered out; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “freq300” means that only dependencies with frequency more than 300 are selected

	P	AR	EM	C	T_P
baseline	0	0	43.76	94.7	11.62
<i>MMMT, 4 votes</i>	96.52	73.51	41.73	90.7	1.19
MMMT, 4 votes, L10	96.56	72.31	41.56	90.8	1.24
MMMT, 4 votes, L5	96.71	69.14	41.11	91.5	1.37
<i>MMMT, 4 votes, pos8</i>	97.84	26.72	43.25	97.6	5.79
MMMT, 4 votes, dep8	97.03	51.50	41.90	92.6	2.11
MMMT, 4 votes, dep10	97.03	57.37	41.45	91.6	1.80
MMMT, 4 votes, freq300, prec70	96.90	69.66	42.29	91.5	1.36
MMMT, 4 votes, dep10, L5	97.09	55.06	41.34	92.0	2.00
MMMT, 4 votes, freq300, prec70, L5	96.97	66.31	42.07	92.0	1.49
MMMT, 4 votes, prec70, L5	96.97	66.31	42.07	92.0	1.49
MMMT, 4 votes, prec80, L5	97.41	61.18	42.07	92.7	1.81
<i>MMMT, 4 votes, prec80, L3</i>	97.46	53.96	42.41	93.4	2.17
MMMT, 4 votes, pos8, L5	97.90	26.10	43.19	97.5	6.04
MMMT, 4 votes, prec70 or pos8, L5	96.95	66.55	42.07	92.0	1.48

Table 6.20: Ensemble MMT, tuning. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “dep10” means that only dependencies of one of the 10 dependency types are selected; “pos10” means that only dependencies for dependent token of one of the 10 PoS are selected; “L5” means that dependencies spanning more than 5 tokens are filtered out; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “freq300” means that only dependencies with frequency more than 300 are selected

	P	AR	EM	C	T_P
baseline	0	0	43.76	94.7	11.62
BMMMT, 3+ votes	92.99	84.94	38.25	74.2	0.90
<i>BMMMT, 5 votes</i>	97.20	72.01	43.36	93.4	1.29
BMMMT, 5 votes, L10	97.24	70.86	43.14	93.5	1.32
BMMMT, 5 votes, L5	97.35	67.81	42.80	93.9	1.46
BMMMT, 5 votes, dep8	97.47	50.72	42.69	94.0	2.22
BMMMT, 5 votes, dep10	97.49	56.46	42.41	93.0	1.93
BMMMT, 5 votes, freq300, prec70	97.50	68.36	43.64	93.8	1.48
BMMMT, 5 votes, dep10, L5	97.53	54.22	42.29	93.4	2.13
BMMMT, 5 votes, freq300, prec70, L5	97.55	65.12	43.42	94.2	1.58
BMMMT, 5 votes, prec70, L5	97.55	65.12	43.42	94.2	1.59
BMMMT, 5 votes, prec80, L5	97.88	60.21	43.42	94.4	1.89
<i>BMMMT, 5 votes, prec80, L3</i>	97.91	53.14	43.59	94.9	2.28
BMMMT, 5 votes, prec70 or pos8, L5	97.55	65.12	43.42	94.2	1.58
<i>BMMMT, 5 votes, dep11</i>	98.24	24.91	44.26	98.1	4.86
BMMMT, 5 votes, dep11, L5	98.40	23.24	44.15	97.8	5.32

Table 6.21: Ensemble BMMMT, tuning. Static evaluation of ensembles before integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using exact match (“EM”), coverage (“C”) and the running time of PET (“ T_P ”). Number of votes specifies how many parsers in ensemble should agree on a labeled dependency; “dep10” means that only dependencies of one of the 10 dependency types are selected; “pos10” means that only dependencies for dependent token of one of the 10 PoS are selected; “L5” means that dependencies spanning more than 5 tokens are filtered out; “prec80” means that only those dependency types were selected for which precision is 80% or higher on the development set for all the parsers in the ensemble; “freq300” means that only dependencies with frequency more than 300 are selected

experiments on the test set the setups (1) MMT with 4 votes; (2) MMT with 4 votes and only 8 PoS from static analysis and (3) MMT with 4 votes, minimum dependency precision 80% and maximum dependency length 3 are chosen as the candidates for the best time, coverage and a compromise between the two.

Table 6.21 visualizes tuning experiments with the BMMMT ensemble. Compared to the BMT and MMT ensembles, the coverage for the BMMMT ensemble with a strict voting scheme is substantially higher from the outset. For all three ensembles we observe a trade-off between the coverage and the annotation rate: the coverage is maximized when the rigorous filtering is applied, and the remaining top-quality dependencies preserved for parser combination constitute roughly a quarter of all dependencies originally generated by the parsers in the ensemble. This in turn has an impact on the final parsing time as the fewer dependency constraints are used, the less speed-up is achieved. From all the possible setups of the BMMMT ensemble we have chosen (1) BMMMT with 5 votes, (2) BMMMT with 5 votes and 11 dependency types and (3) BMMMT with 5 votes, minimum dependency precision of 80% from the static analysis and maximum length 3 guided by the same criteria as for other ensembles: best speed, coverage

	LAS	UAS	LACC	Time, self-eval.	Time, average
Malt	88.83	91.06	91.1	7.201	0.0048787
MST	90.23	92.70	92.29	330.438	0.223874
B&N with confidence scoring	92.11	94.19	93.78	1242	0.8415
Mate	92.17	94.13	93.86	269.1	0.1823
Turbo	91.62	93.81	93.37	89.935	0.0609
MST, KD-Fix*0.05*50	90.23	92.70	92.29	6678.886	4.525
MST, KD-Fix*0.05*100	90.23	92.70	92.29	11254.396	7.6249

Table 6.22: Evaluation of individual parsers including punctuation on the test set (section 21 of DeepBank). Self-evaluation time is measured in seconds, average time per sentence is measured in seconds per sentence. KD-Fix*0.05*50 stands for the KDF-Fix algorithm with the parameters standard deviation of 0.05 and K of 50

and some middle ground correspondingly.

6.6 In-domain parser integration experiments

In this section we will discuss the experiments performed on the test set of DeepBank (the section 21) with the various setups that we have selected during the tuning phase. The goal of these experiments is to achieve a speed-up of the HPSG parser without significant loss of coverage and accuracy.

The performance of the individual parsers on the test set is shown in Table 6.22. The experiments are performed on a powerful computing cluster with all nodes having a minimum 64 GB RAM, 16 physical CPU cores and connected by FDR (56 Gbps) Infiniband whereas for the tuning phase the corresponding evaluations presented in Table 6.4 used an older and less powerful computer cluster. For B&N a more powerful computing cluster ensures significantly better running time. In all experiments in this section B&N is run with parameters to produce per dependency scores though the confidence scores are only used for confidence thresholding with B&N and are not employed in the ensemble methods. Similarly as during tuning, efficiency is estimated allowing a dependency parser to use only one CPU. We assume that individual parsers in an ensemble can be run in parallel and that the total effective running time of the ensemble is the performance of the slowest parser.

Table 6.22 demonstrates that the newer parsers, i.e. B&N, Mate and Turbo, are more accurate than Malt and MST. Malt is, however, the fastest, followed by Turbo which is 12 times slower, and B&N is the slowest among the five. It is interesting to note that MST is 20 times slower when it is run with the activated KD-Fix algorithm and the K-parameter set to 50; when K is increased to 100, the difference in running time is 34 times.

Selected setups from the tuning phase are re-trained with the released versions of the ERG 1214 and DeepBank 1.1 and evaluated on the test set of the treebank. An overview of the results is given in Table 6.23. As we recall from the static analysis defined in Section 6.4, precision (“P”) shows how accurate the selected dependencies are and the annotation rate (“AR”) indicates how many dependencies were actually chosen for combination. The standard metrics in

	P	AR	EM	C	T_D	T_P	T_T
baseline	0	0	39.81	95.4	0	8.16	8.16
upperbound	100	85.87	68.62*	100*	0	0.82	0.82
Turbo, dep10, pos10, L5	96.22	29.24	37.02*	94.9*	0.06	3.88	3.94
MST, KD-Fix*0.05*50, thr1, pos8	98.59	20.73	39.17	97.4*	4.53	5.74	10.27
MST, KD-Fix*0.05*50, thr1, prec80, l5	97.97	44.19	37.88	94.3	4.53	3.23	7.76
MST, KD-Fix*0.05*100, thr1, dep7	98.96	15.73	39.74	97.1*	7.62	5.91	13.53
<i>B&N, prec95, PD20</i>	97.11	8.25	39.03	95.9	0.84	6.66	7.50
B&N, prec95, PD40	95.97	31.25	37.53*	94.2	0.84	2.76	3.60
<i>BMT, 3 votes, pos8</i>	97.67	26.96	38.96	97.6*	0.84	4.49	5.33
BMT, 3 votes, -prec<80, L5	97.42	62.86	38.74	93.0	0.84	1.50	2.34
BMT, 3 votes	96.28	76.96	38.53	90.2*	0.84	1.00	1.84
<i>MMMT, 4 votes, pos8</i>	97.97	26.42	39.10	98.1*	0.22	4.61	4.83
MMMT, 4 votes, -prec<80, L3	97.70	54.07	38.88	94.4	0.22	2.02	2.24
MMMT, 4 votes	96.76	72.63	38.67	91.6*	0.22	1.17	1.39
<i>BMMMT, 5 votes, dep11</i>	98.16	25.24	39.89	97.4*	0.84	3.97	4.81
<i>BMMMT, 5 votes, -prec<80, L3</i>	98.14	53.27	39.39	95.9	0.84	2.11	2.95
<i>BMMMT, 5 votes</i>	97.42	71.23	40.03	94.5	0.84	1.23	2.07

Table 6.23: In-domain parser combination experiments. “ T_D ” - running time of a dependency parser/ensemble, “ T_P ” - running time of PET with dependency restrictions, “ T_T ” - total running time of parser integration setup. Statistically significant results for the exact match and coverage are marked with an *. KD-Fix*0.05*50 stands for the KDF-Fix algorithm with the parameters standard deviation of 0.05 and K of 50, “thr” stands for the threshold on per-dependency probability score; “dep7” means that only dependencies of one of the 7 dependency types are selected; “-prec<80” means that dependency types that had precision less than 80% on the development set for at least one of the parsers in the ensemble are filtered out; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

ERG parsing, exact match (“EM”) and coverage (“C”), describe PET’s precision and robustness. Since the efficiency of the dependency parsers and PET with dependency restrictions were evaluated on the same computer cluster, we are approximating the running time of parser integration: “ T_D ” measures how much time it takes to produce bilinear dependencies, “ T_P ” is the running time of PET with dependency restrictions, and “ T_T ” is the total running time of parser integration, i.e. the sum of “ T_D ” and “ T_P ” (all three measures are expressed in seconds per sentence).

In the same manner as during tuning, we introduce a baseline by running PET without dependency constraints with a maximum processing time per sentence of 5 minutes and we establish the upper bound by running PET with gold dependencies. Compared to the baseline, the upper bound parser has 1.7 times higher exact match, 4.6 percentage points higher coverage and it is almost 10 times faster. Our primary goal is to achieve a speed-up over the baseline while not compromising baseline coverage and minimizing the loss in exact match.

In the first parser combination experiment we generate dependencies with Turbo and apply

the following filtering pattern: select a dependency only if it belongs to one of the 10 dependency types, spans over less than 5 tokens and has a dependent from one of the 10 PoS types. Static analysis shows that with this filtering policy we use only 29.24% of the dependencies generated by Turbo with an average precision of 96.22%. We find that parser integration in this setup is twice as fast as the baseline, but the exact match and coverage are significantly lower than the baseline.

In the second set of parser combination experiments we test three setups with the MST parser with the KD-Fix algorithm: (1) MST with the KD-Fix algorithm with the standard deviation of 0.05 and K equals 50, selecting only dependencies with confidence score 1 and with an application of a filtering parameter by PoS tag: only dependencies with the dependent of 8 specific PoS tags are selected; (2) similar to the first one, but with different filtering parameters: only dependencies spanning less than five tokens and belonging to a dependency type that had precision of 80% or more on the development set during tuning are selected; (3) MST with the KD-Fix algorithm with the standard deviation of 0.05 and K equals 100, selecting only dependencies with the confidence score of 1 and only of 7 dependency types. As we can see from Table 6.23, the first and the third setups are not interesting because there are no efficiency gains compared to the baseline. Despite a minor 0.4 second speed-up, the second setup is also not very promising because of the drop in exact match and coverage, although the statistical significance tests do not confirm that the difference from the baseline is significant.

In the third set of parser combination experiments we integrate PET with B&N with different confidence thresholds tuned on the development set using precision and proportion of selected dependencies PD (which is computed including punctuation, multiword expressions and contracted negation as explained in the previous section). The setups differ in the quantity of chosen dependencies: for the first one annotation rate is only 8.25% and for the second one annotation rate is 31.25%. The first setup might be considered interesting as it offers some 0.5 seconds speed-up while the coverage is on the baseline level and the loss of exact match is insignificant. We further find that although the second setup is rather fast (3.60 seconds in contrast with 8.16 seconds baseline), the exact match is insufficient.

The fourth block of experiments concerns the parsing ensemble of B&N, Mate and Turbo (BMT) in three configurations: (1) filtering by the PoS type of dependent; (2) filtering by dependency type and dependency length; (3) no filtering. For all the experiments with ensembles, we consider efficiency of the ensemble equal to the efficiency of the slowest parser (in this case B&N with the processing time 0.84 seconds per sentence) because parsers in the ensemble can be run in parallel. Although the first configuration has the lowest annotation rate (26.96%), it is the most promising of all three configurations: we observe improvements in time and coverage over the baseline with insignificant loss in exact match. The other two configurations exhibit a speed-up but at a cost of low coverage. Whilst the drop in coverage is significant for the third setup, the significance tests are insensitive to the difference between the baseline coverage and the coverage of the second setup.

The fifth parser integration experiment is realized with an ensemble of Malt, MST, Mate and Turbo (MMMT) in three configurations analogous to the BMT ensemble, with an exception that in the second configuration the imposed restrictions on the length of dependencies are tighter: less than three tokens compared to less than five for the BMT ensemble. MST is the slowest parser in this ensemble therefore its efficiency, 0.22 seconds per sentence, is used to evaluate

	Self-evaluation time, average, sent per sec			
	CB	VM	SC	WS
Malt	0.01	0.005	0.007	0.011
MST	0.36	0.04	0.09	0.12
B&N	1.88	0.22	1.29	2.56
Mate	0.18	0.07	0.26	0.22
Turbo	0.12	0.05	0.08	0.18
MST, KD-Fix*0.05*50	3.92	0.75	2.99	3.22

Table 6.24: Evaluation of individual parsers including punctuation on CB, VM, SC and WS). Self-evaluation time is measured in seconds, average time per sentence is measured in seconds per sentence

the efficiency of the ensemble. Similarly as for the BMT ensemble, the first configuration with filtering by the PoS type of the dependent is the best: both time and coverage improve and exact match remains close to the baseline. The other two configurations result in a loss of coverage, significant for the third setup and possibly due to chance for the second setup.

The last set of experiments is with an ensemble of B&N, Malt, MST, Mate and Turbo (BMMMT) with the time 0.84 seconds per sentence equal to the slowest parser in the ensemble, i.e. B&N. In the first configuration the filtering criterion is dependency type, in the second configuration the filtering is realized by another dependency type criterion and dependency length; in the third configuration no filtering is applied. The most successful setup is the first one as it delivers good improvements of coverage and time and baseline value of exact match. The second and third configurations are quite good as they are twice as fast as the first one and do not cause a decrease of the exact match, but they do not, however, improve the coverage over the baseline.

The overall best configurations are highlighted in italics and bold fonts in Table 6.23. These are setups that allow us to achieve the main goal: to substantially reduce running time and in some cases also coverage with minimal or no loss of exact match. In all experiments we observe tradeoffs in coverage and efficiency: improving the running time of the integrated setup we risk compromising coverage. We conclude that the ensemble approach to selective pruning of dependencies seems to be most promising, and it appears that the more parsers are used for the voting strategy, the better results are achieved.

6.7 Cross-domain parser integration experiments

In Chapter 5 we contrasted in-domain and cross-domain performance of different parsers and observed that the grammar-based parser seems to be more domain-resilient. In this section we investigate whether the efficiency gains obtained with the parser integration setups on the in-domain data almost without loss of coverage and exact match, will generalize to the out-of-domain data. For this purpose we use the same datasets from the Redwoods treebank as in Section 5.5: CB, SC, VM and WS. The domain VM requires a special “speech” version of ERG.

	P	AR	EM	C	T _D	T _P	T _T
baseline	0	0	29.53	97.0	0	6.00	6.00
upperbound	100	87.17	65.77*	100*	0	0.73	0.73
Turbo, dep10, pos10, L5	92.67	25.69	27.01	90.6*	0.12	3.17	3.29
B&N, prec95, PD20	95.21	6.66	30.37	96.8	1.88	4.95	6.83
B&N, prec95, PD40	92.53	29.36	28.19	86.2*	1.88	1.95	3.83
<i>BMT, 3 votes, pos8</i>	95.65	21.85	29.53	96.8	1.88	3.71	5.59
BMT, 3 votes, -prec<80, L5	94.43	62.03	27.85	81.2*	1.88	1.14	3.02
BMT, 3 votes	93.04	72.82	27.01	76.8*	1.88	0.87	2.75
<i>MMMT, 4 votes, pos8</i>	95.48	21.35	28.69	96.6	0.36	3.91	4.27
MMMT, 4 votes, -prec<80, L3	93.83	53.01	26.68	79.2*	0.36	1.39	1.75
MMMT, 4 votes	92.72	68.51	26.01	73.8*	0.36	1.03	1.39
<i>BMMMT, 5 votes, dep11</i>	96.82	22.81	29.53	98.0	1.88	3.10	4.98
BMMMT, 5 votes, -prec<80, L3	95.68	51.00	28.86	89.9*	1.88	1.51	3.39
BMMMT, 5 votes	94.92	65.39	28.36	84.9*	1.88	1.09	2.97

Table 6.25: Out-of-domain parser combination experiments on **CB**. The static evaluation of ensembles before the integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using the exact match (“EM”), the coverage (“C”), the running time of a dependency parser/ensemble (“T_D”), the running time of PET with dependency constraints (“T_P”) and the total running time of a parser integration setup (“T_T”). Statistically significant results for the exact match and coverage are marked with an *. The number of votes specifies how many parsers in an ensemble should agree on a labeled dependency; “prec95”, “PD20” means that the values of the e-, x- and a-scores of B&N are tuned so that at least 20% of all the generated dependencies are selected for the integration with PET and the accuracy of the selected dependencies is at least 95%; “dep7” means that only dependencies of one of the 7 dependency types are selected; “-prec<80” means that dependency types that had precision less than 80% on the development set for at least one of the parsers in the ensemble are filtered out; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

The individual running times of the dependency parsers are displayed in Table 6.24. B&N time is measured with confidence scoring parameters. We do not include experiments with the confidence scores produced by the MST parser on the out-of-domain data since the total running time of the integrated setups on the in-domain data is below the baseline and efficiency on the out-of-domain data is also very low as follows from Table 6.24.

Table 6.25 describes the experiments on the **CB** test set. As usual in the present study, the baseline is established by parsing with PET without dependency constraints, and the upper bound is the PET parser run with gold dependencies. The baseline values for the exact match is 27.01%, for coverage 90.6% and for the total running time 6 seconds per sentence. The upper bound for the exact match is 65.77%, for coverage 100% and for the total running time 0.73 seconds per sentence.

Despite the improvements in the total running time, most of the setups have insufficient coverage on the **CB** set. However, B&N with confidence thresholding parameters selected by

	P	AR	EM	C	T _D	T _P	T _T
baseline	0	0	48.22	99.8	0	0.78	0.78
upperbound	100	82.89	71.98*	100	0	0.19	0.19
Turbo, dep10, pos10, L5	92.43	31.77	46.50	92.6*	0.05	0.43	0.48
B&N, prec95, PD20	92.21	6.18	47.72	99.0	0.22	0.70	0.92
B&N, prec95, PD40	88.73	23.80	44.97*	93.2*	0.22	0.38	0.60
BMT, 3 votes, pos8	96.48	26.04	47.21	98.0*	0.22	0.52	0.74
BMT, 3 votes, -prec<80, L5	92.93	59.77	44.06*	88.3*	0.22	0.31	0.53
BMT, 3 votes	91.25	65.97	43.55*	86.7*	0.22	0.25	0.47
MMMT, 4 votes, pos8	96.24	25.58	47.92	97.9*	0.07	0.51	0.58
MMMT, 4 votes, -prec<80, L3	94.23	52.81	48.63	93.7*	0.07	0.36	0.43
MMMT, 4 votes	92.11	61.14	46.60	89.2*	0.07	0.26	0.33
BMMMT, 5 votes, dep11	95.07	23.73	47.11	96.6*	0.22	0.57	0.79
BMMMT, 5 votes, -prec<80, L3	95.42	50.79	48.83	95.6*	0.22	0.37	0.59
BMMMT, 5 votes	93.72	58.16	47.01	92.6*	0.22	0.29	0.51

Table 6.26: Out-of-domain parser combination experiments on VM. The static evaluation of ensembles before the integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using the exact match (“EM”), the coverage (“C”), the running time of a dependency parser/ensemble (“T_D”), the running time of PET with dependency constraints (“T_P”) and the total running time of a parser integration setup (“T_T”). Statistically significant results for the exact match and coverage are marked with an *. The number of votes specifies how many parsers in an ensemble should agree on a labeled dependency; “prec95”, “PD20” means that the values of the e-, x- and a-scores of B&N are tuned so that at least 20% of all the generated dependencies are selected for the integration with PET and the accuracy of the selected dependencies is at least 95%; “dep7” means that only dependencies of one of the 7 dependency types are selected; “-prec<80” means that dependency types that had precision less than 80% on the development set for at least one of the parsers in the ensemble are filtered out; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

requiring the parser to achieve a precision of 95% and PD of 20%, shows an improved coverage and a baseline level of the exact match but it is slower than the baseline and therefore deemed as not interesting.

The parser ensemble setups maximized for coverage during tuning (i.e. BMT and MMT with part-of-speech filtering parameters and BMMMT with filtering by dependency type), deliver the best results on CB: improved running time, and coverage and exact match at the baseline level.

None of the setups delivers a speed-up without significant loss in coverage on VM (see Table 6.26). On SC and WS (see Tables 6.27 and 6.28) we obtain acceptable coverage, exact match and time for the BMMMT ensemble with filtering by dependency type and the MMT ensemble with filtering by part-of-speech correspondingly.

During tuning and testing both on the in- and out-of-domain data we observe an inverse relationship between time and coverage. On CB, SC and WS we furthermore notice that within

	P	AR	EM	C	T _D	T _P	T _T
baseline	0	0	38.26	98.7	0	3.47	3.47
upperbound	100	87.29	69.53*	99.9	0	0.61	0.61
Turbo, dep10, pos10, L5	93.53	27.64	35.93	91.5*	0.08	1.80	1.88
B&N, prec95, PD20	94.70	6.61	37.44	97.9	1.29	2.95	4.24
B&N, prec95, PD40	92.09	28.37	35.93	87.2*	1.29	1.40	2.69
BMT, 3 votes, pos8	95.83	24.11	37.67	97.1	1.29	2.22	3.51
BMT, 3 votes, -prec<80, L5	94.30	61.84	35.23	82.8*	1.29	0.93	2.22
BMT, 3 votes	92.50	72.83	32.91*	79.3*	1.29	0.75	2.04
MMMT, 4 votes, pos8	95.89	23.82	37.67	96.5*	0.26	2.19	2.45
MMMT, 4 votes, -prec<80, L3	94.50	54.77	35.00	83.7*	0.26	1.03	1.29
MMMT, 4 votes	92.99	68.64	32.44*	78.8*	0.26	0.79	1.05
<i>BMMMT, 5 votes, dep11</i>	96.47	25.33	37.67	97.1	1.29	1.90	3.19
BMMMT, 5 votes, -prec<80, L3	95.59	52.9	36.98	89.7*	1.29	1.09	2.38
BMMMT, 5 votes	94.33	65.69	34.53*	85.5*	1.29	0.85	2.14

Table 6.27: Out-of-domain parser combination experiments on **SC**. The static evaluation of ensembles before the integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using the exact match (“EM”), the coverage (“C”), the running time of a dependency parser/ensemble (“T_D”), the running time of PET with dependency constraints (“T_P”) and the total running time of a parser integration setup (“T_T”). Statistically significant results for the exact match and coverage are marked with an *. The number of votes specifies how many parsers in an ensemble should agree on a labeled dependency; “prec95”, “PD20” means that the values of the e-, x- and a-scores of B&N are tuned so that at least 20% of all the generated dependencies are selected for the integration with PET and the accuracy of the selected dependencies is at least 95%; “dep7” means that only dependencies of one of the 7 dependency types are selected; “-prec<80” means that dependency types that had precision less than 80% on the development set for at least one of the parsers in the ensemble are filtered out; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

	P	AR	EM	C	T _D	T _P	T _T
baseline	0	0	38.34	97.5	0	5.05	5.05
upperbound	100	86.1	69.96*	100*	0	0.65	0.65
Turbo, dep10, pos10, L5	89.52	25.32	35.43	89.7*	0.18	2.36	2.54
B&N, prec95, PD20	95.55	5.67	38.12	97.3	2.56	4.16	6.72
B&N, prec95, PD40	92.71	29.00	35.20	87.2*	2.56	1.95	4.51
BMT, 3 votes, pos8	92.12	19.96	37.67	96.6	2.56	2.89	5.45
BMT, 3 votes, -prec<80, L5	94.18	62.16	36.32	85.9*	2.56	0.98	3.54
BMT, 3 votes	92.84	73.69	34.08	83.4*	2.56	0.77	3.33
MMMT, 4 votes, pos8	91.33	19.95	37.22	95.5	0.22	2.92	3.14
MMMT, 4 votes, -prec<80, L3	93.44	54.16	32.96*	80.3*	0.22	1.13	1.35
MMMT, 4 votes	91.85	70.93	32.06*	76.7*	0.22	0.83	1.05
BMMMT, 5 votes, dep11	95.59	21.79	36.32	94.8	2.56	2.72	5.28
BMMMT, 5 votes, -prec<80, L3	95.48	51.84	35.65	88.1*	2.56	1.23	3.79
BMMMT, 5 votes	94.23	67.24	35.20	86.3*	2.56	0.89	3.45

Table 6.28: Out-of-domain parser combination experiments on *WS*. The static evaluation of ensembles before the integration is carried out using precision (“P”) and annotation rate (“AR”); setups integrated with PET are evaluated using the exact match (“EM”), the coverage (“C”), the running time of a dependency parser/ensemble (“T_D”), the running time of PET with dependency constraints (“T_P”) and the total running time of a parser integration setup (“T_T”). Statistically significant results for the exact match and coverage are marked with an *. The number of votes specifies how many parsers in an ensemble should agree on a labeled dependency; “prec95”, “PD20” means that the values of the e-, x- and a-scores of B&N are tuned so that at least 20% of all the generated dependencies are selected for the integration with PET and the accuracy of the selected dependencies is at least 95%; “dep7” means that only dependencies of one of the 7 dependency types are selected; “-prec<80” means that dependency types that had precision less than 80% on the development set for at least one of the parsers in the ensemble are filtered out; “L5” means that dependencies spanning more than 5 tokens are filtered out; “pos8” means that only dependencies for dependent token of one of the 8 PoS are selected

ensemble groups higher coverage comes in pair with higher exact match. The latter however does not hold for VM: for the MMMT ensembles, for example, the slowest setup with a total running time of 0.58 seconds per sentence has a coverage of 97.9% and an exact match of 47.92% while a slightly faster setup with a total running time of 0.43 seconds per sentence has a coverage of 93.7% and an exact match of 48.63%.

In conclusion, the out-of-domain parsing is generally faster than the in-domain parsing in our experiments, which at least in part reflects shorter average sentence length and, thus, lower structural complexity in our out-of-domain data, and the loss in coverage is larger. The MMMT ensemble with filtering by part-of-speech and the BMMMT ensemble with filtering by dependency type are the most robust across domains.

6.8 Summary

The goal of the present chapter is to explore the space of parser combination setups in order to achieve efficiency gains over the baseline for grammar-based parsing without loss in coverage and accuracy. Bilexical dependencies serve as an interface for the integration.

The main challenge of successful parser combination is to choose only high-quality dependencies. The tested approaches include filtering, confidence thresholding and ensembles with voting scheme. The confidence thresholding method did not deliver the first-rate results. For B&N, per-dependency confidence scores do not correlate well with the correctness of the attachment and labeling. For this reason we manage to select only a small number of reliable dependencies which is in most setups not enough to improve efficiency of the HPSG parsing. For MST, the parsing time increases dramatically when the algorithm for generation of per-dependency probability scores is activated which makes it unattractive candidate for parser combination. Ensemble method in combination with filtering, on the other hand, is fast and reliable in picking out accurate dependencies.

We find that the results of our parser combination experiments are slightly better than one could have expected both in- and out-of-domain as there are setups that allow us to realize our goal of improving efficiency of PET. On the in-domain data not only we achieve a speed-up, but in some configurations we also significantly improve coverage without causing a decrease in accuracy. However, we did not achieve any gains in the exact match with the parsing combination approach. Ensembles seem to be more resilient to domain shift than individual parsers as the best results are achieved with MMMT with part-of-speech filtering and BMMMT with filtering by dependency type.

Chapter 7

Conclusion

In the present work we have sought to bridge the gap between grammar-based and data-driven parsing using bilexical dependencies as an intermedium. The primary aims of this dissertation have been contributing to the understanding of the range of various dependency formats, comparing a hybrid “deep” grammar-based parser to state-of-the-art statistical data-driven parsers and improving the efficiency of the grammar-based parser using bilexical dependency constraints.

7.1 Main conclusions

In the following we provide a summary of the project components and highlight the main findings and empirical results.

In Chapter 3 we propose a conversion procedure from representations of the English Resource Grammar (Flickinger, 2000) to bilexical dependencies in order to prepare a basis for parser comparison and parser combination and offer a more conventional format for the pre-existing DELPH-IN treebanks for statistical parsing. Syntactic dependencies are obtained by a procedure similar to phrase structure tree transformation to dependencies, i.e. by traversing the derivation tree of the ERG and looking up the head daughter for each branching rule in the grammar specification file. The resulting dependency format is projective. On the semantic level conversion is performed from a reduced representation of MRS (Copestake et al., 2005)—so-called Elementary Dependency Structures (Oepen and Lønning, 2006)—that discards scope-related information. The converter itself is now freely available from the repository of DELPH-IN resources.

The proposed dependency representations offer a dependency format for researchers outside the DELPH-IN community who might be interested in using DELPH-IN treebanks, thus both DT and DM are included in the standard release of DeepBank. DM graphs were used as one of three target semantic representations in the Broad-Coverage Semantic Dependency Parsing tasks in 2014 and 2015 (Oepen et al., 2014, 2015).

Formal representation of syntax is continuously debated and there exists a variety of annotation schemes. Variation in dependency annotation is partially related to conflicting criteria for identifying the head. We provide an overview of several syntactico-semantic dependency representations (CD, CP, SB, SD, EP, PA, PT, DT and DM), carry out a detailed contrastive study

of different choices made in different formalisms, such as functional or substantive heads, and graph structural properties such as connectivity. For example, formats that are more semantic in spirit usually exclude semantically vacuous words from the dependency graph, which also affects the graph structure: analyses assigned to a sentence by more semantic formats often do not constitute an acyclic tree structure as opposed to more syntactic representations. For many linguistic constructions including coordination and prepositional phrases there is disagreement among the formats about the representation of dependency structure. Quantitative comparison also indicates relatively low structural similarity of the formats in terms of Jaccard and F1 scores. The Stanford formats are most related to each other, PA and DT have highest pairwise similarity scores with CD, DM shows agreement with EP, and PT corresponds with SD.

In Chapter 4 we further compare syntactic dependency formats intrinsically using three direct data-driven dependency parsers and extrinsically in application to negation resolution, in order to investigate which properties of an annotation scheme make it difficult to parse and what effects the choice of the dependency format has on a downstream application.

As the transition-based and graph-based approaches have dominated the field of dependency parsing, we have chosen for experimentation the transition-based parser Malt, the graph-based parser MST and the Bohnet and Nivre (2012) parser (B&N) that brings together a transition-based approach with global modeling. We use the syntactic annotation formats SB, CD, DT and PA in combination with two PoS tag sets: PTB tags and supertags (more fine-grained lexical types of the ERG). It follows from the results that B&N has the highest labeled accuracy among the three parsers for most of the setups, and MST outperforms Malt in the conventional setup with PTB PoS tags on all the formats. We find that SB is the easiest format to label and CD is the easiest to process structurally, while PA appears to be the hardest annotation scheme for all three parsers. DT is more difficult than SB and CD for Malt and MST, though B&N is able to learn DT as well as SB and CD. As expected, there is a strong correlation between the PoS tag sets and the annotation schemes: PTB tags suit the PTB-derived SB, CD and PA formats well and supertags fit the DeepBank-derived DT format better. There is a many-to-many correspondence between PTB tags and supertags which indicates that supertags do not provide an extra level of detail over PTB tags but rather complement them. Punctuation marks are treated as postfixes in ERG and as a consequence punctuation signs in DT are always attached to the nearest token to the left which is an easy rule for a parser to learn in contrast to complex punctuation attachment rules in the PTB-derived SB, CD and PA formats.

In the error analysis of the parsing results we examined LAS over individual PoS and discussed problems of parsing coordination and some verb types. DT and PA take the lowest-accuracy conjunction-headed approach to coordination analysis which however disambiguates cases of individual and shared modifiers as in the example “*old men and women*”. The verb errors typically occur in longer sentences on dependencies spanning over many words and are frequently related to incorrect root identification. Common errors of parsing the PA format are related to the inconsistent analysis of phrases with dollar sign and numbers such as “\$ 3 billion” which are very frequent in WSJ text, complex punctuation attachment and overloaded labels.

The extrinsic evaluation of dependency formats on the negation resolution task shows that CD is best for detection of negated events, SB for scopes and DT for scope tokens, global negation and correct negated sentences. Both parsing and negation resolution results provide convincing evidence that despite structural differences, DT is in many aspects similar to SB and

CD and that these three schemes are to some extent interchangeable.

In Chapter 5 we use DT dependencies as the basis for cross-framework parser comparison. We seek to measure the differences in accuracy, coverage and efficiency of grammar-based and data-driven phrase structure and dependency syntactic analyzers and document the effects of domain variation. We evaluate in-domain trade-offs in coverage, time, tagging accuracy and labeled and unlabeled dependency accuracies for the PET (Callmeier, 2000), B&N (Bohnet and Nivre, 2012) and Berkeley (Petrov et al., 2006) parsers. We observe that in terms of coverage the grammar-driven parser is lagging behind the data-driven systems with two sources of incomplete analysis: parse failure and tokenization mismatch against the gold standard. The second problem is caused by a technical requirement to start processing from a raw string while the statistical systems accept pre-tokenized input. The parsing time of the grammar-based parser in a configuration optimized for efficiency using the joint segmentation and supertagging setup of Dridan (2013a) is comparable with B&N but stands behind Berkeley. The main result, which also holds in cross-domain experiments, concerns significantly higher tagging and dependency accuracies of the grammar-based parser in contrast to the data-driven systems.

We further carry out a set of sanity experiments where we show that the achieved results are not owed to the use of complex lexical types and an alternative tokenization approach. In the error analysis we find that head-complement and head-specifier are the easiest relations and head-adjunct is the most difficult relation for all three parsers. B&N and Berkeley perform similarly on short dependencies, and B&N is slightly more accurate than Berkeley on longer dependency relations, while PET systematically outperforms the two other parsers in terms of precision and recall of dependencies spanning different number of tokens. The systems generalize well for nouns and verbs and make most errors on conjunction structures and prepositions. The downstream tasks of negation resolution and semantic dependency parsing are used to test whether observed intrinsic differences in syntactic parsing performance are significant for applications that exploit syntactic features. The results are ambivalent as for the first task we get a negative answer and for the second task we get a positive answer. This may mean that some applications are less sensitive than others to the accuracy of syntactic processing.

In different subfields of natural language processing a combination of different approaches have facilitated improved results. In this dissertation we aim to bring together grammar-based and data-driven dependency parsing. The strengths and differences of the two approaches suggest that there are good grounds for combination of fast and robust statistical data-driven systems deriving analyses based on rich feature representations and scalable machine learning algorithms, on the one hand, with grammar-based systems enhanced with statistical modules for disambiguation producing theoretically-motivated fine-grained analyses that account for subcategorization, multiword expressions and other subtle linguistics distinctions, on the other hand. Carrying out a set of parser combination experiments in Chapter 6, we observe that the evaluation metrics of efficiency and coverage stand in an inverse relationship, by improving one of these two metrics, we risk to decrease the other one. Similarly, we observe trade-offs of accuracy and efficiency: for the faster parser combination setups the exact match metric usually shows a lower value. The relationship between accuracy and coverage is less clear: improved coverage does not always lead to improved exact match, however poor coverage usually results in low exact match. Exploring the space of various integrated setups we search for the ones that offer an increase of efficiency over the baseline without sacrificing coverage or accuracy.

We seek to improve the efficiency of PET by constraining the application of linguistically motivated wide-coverage HPSG rules with bilexical dependencies produced by data-driven dependency parsers. We propose two methods for selecting high-quality dependencies generated by data-driven parsers: relying on the per-dependency confidence scores produced by the individual parsers and applying several filters—pruning dependencies by length, type and part-of-speech tag of the node. In addition, we expand the ensemble approach of Sagae et al. (2007) by including more state-of-the-art statistical parsers into an ensemble and applying our filtering method to the output of the voting. We introduce a “static” analysis to reduce the space of candidate combination methods. For most of the setups we present three configurations optimized by different criteria: with maximized coverage, with minimized running time, with balanced improvements in coverage and efficiency. Further experiments on the out-of-domain data indicate that there are configurations that achieve efficiency gains which are resilient across a number of domains with minimal loss in coverage and accuracy.

7.2 Research questions revisited

In the introduction of the thesis we posed several research questions which we will re-visit here and briefly outline the results our study has yielded.

- *Which abstract commonalities and differences can be identified among various dependency representations?*

At the start of our project in 2011, we witnessed a variety of dependency frameworks with no unique standard. In our work we survey several syntactico-semantic dependency schemes for English, focusing on structural differences and similarities. We observe a striking disagreement among the formats with respect to graph structure, head properties and pairwise similarity. For a number of linguistic constructions such as coordination, prepositional phrases, complex verbs, complex nouns we observe a lot of variation in annotation. Some formats are designed to ease computation while others tend to offer more linguistically-motivated analysis. From quantitative analysis, pairwise Jaccard similarities and F1 scores are comparatively low for the different formats. For our ERG-derived representations we discover that DT correlates well with CD and DM is related to EP.

- *How and to what degree can the syntactic and semantic layers of HPSG be expressed in the form of bilexical dependencies?*

We propose a deterministic conversion procedure in the spirit of Zhang and Wang (2009) from the HPSG derivation tree to syntactic bilexical dependencies that we name DT and from elementary dependency structures (Oepen and Lønning, 2006) - a reduced representation of Minimal Recursion Semantics (Copestake et al., 2005) - to semantic bilexical dependencies dubbed DM, and show that the former scheme is most similar to CoNLL (Johansson and Nugues, 2007) among other syntactic formats, and the latter one—to Enju predicate-argument structures (Miyao and Tsujii, 2005) among other semantic formats.

- *How does the choice of syntactic dependency annotation format affect the performance of dependency parsers and downstream applications?*

We compare how well three statistical data-driven parsers learn four syntactic dependency formats from annotated data with the goal to better understand the effect that choice of representation has on the performance of dependency parsers. We find that for Malt (Nivre et al., 2007b) and MST (McDonald et al., 2005b) the linguistically-motivated DT and PA (Čmejrek et al., 2004) formats appear to be harder to learn than the CD (Johansson and Nugues, 2007) and SB (de Marneffe et al., 2006) formats which are custom-designed for dependency parsing, however, the performance of B&N (Bohnet and Nivre, 2012) is not significantly affected by the choice between the CD, SB and DT annotation schemes. We observe that most difficult structures for an automatic analysis include coordinations, preposition phrases and some verbal constructions. In an extrinsic evaluation of the negation resolution task each of the three dependency representations optimize different evaluation metrics with DT showing best results for such metrics as scope tokens, global negation and correct negated sentence.

- *How does the performance of the HPSG parser relate to data-driven syntactic analyzers?*

We perform a comprehensive comparison of a grammar-based system to data-driven syntactic analyzers on the basis of a shared dependency representation and measure parser performance with respect to several aspects: PoS tagging accuracy, coverage, efficiency, labeled and unlabeled accuracy, domain variation and in application to the downstream tasks of negation resolution and semantic dependency parsing. The results indicate that deep grammar parsing yields superior accuracy and better domain resilience than purely statistical systems, though 15% of the in-domain data is outside the grammar coverage and might in theory pose a tougher challenge for grammar-based than data-driven approaches. Extrinsic evaluation highlights different sensitivity levels of applications relying on syntactic features to the choice of the syntactic parsing platform.

- *How can the performance of the HPSG parser be improved by reducing its search space with a native dependency parser?*

We carry out a range of parser combination experiments and show that the HSPG parser integrated with bilexical dependency constraints produced by ensembles of dependency parsers achieves a speed-up at a minimal cost of coverage and accuracy. For some of the setups, the coverage and accuracy are improved over the baseline along with the efficiency in the in-domain experiments. The best results are achieved when only high-precision dependencies are selected for combination. We propose and experiment with three methods and their combination for acquiring most accurate bilexical dependencies from the output of data-driven dependency parsers.

7.3 Future research

There is still a number of paths to explore and open issues to investigate. Below we present a brief overview of extensions and research directions that are potentially interesting for future work.

Conversion of ERG structures to bilexical dependencies In our conversion procedure from syntactic and semantic HPSG structures to bilexical dependencies we have chosen to maximally preserve decisions made in the grammar and therefore produced two novel dependency formats—DT and DM.

The DM format was compared to other semantic dependency formalisms and it was adopted in the shared tasks on Broad-Coverage Semantic Dependency Parsing (Oepen et al., 2014, 2015); nevertheless its properties are not yet sufficiently studied. For example, we are interested to investigate in more detail what information encompassed in MRS structures is lost during the reduction to DM, in particular, what information is expressed by the non-lexical elements from MRS that are excluded during the conversion. Furthermore, the role labels in DM (ARG_1, \dots, ARG_n) could be linked to semantic roles from PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) and the format could be subsequently applied to the task of semantic role labeling.

As a topic for future work we suggest mapping ERG analyses to more standard dependency schemes. Our study indicates that CD and EP are good candidates for mapping because these two formats are structurally the closest to DT and DM correspondingly. Eskelund (2014) designed and implemented a heuristic converter which made use of machine-learned classifiers for labeling, to transform the DT format to CD. The conversion software achieves an unlabeled attachment score of 90.0 and a labeled attachment score of 82.9, which is below the level we would like to reach. We therefore propose to implement conversion directly from the ERG derivation tree to CD and from MRS to EP using additional information encoded in the grammar, rather than using the lossy reductions to DT and DM as intermediate steps in the conversion process.

An alternative direction is mapping ERG analyses to Universal Dependencies (UD) (de Marneffe et al., 2014; Nivre, 2015), a format aiming at consistency in annotation between languages proposed about two years after publication of our work on syntactico-semantic dependencies hence unknown to us at the outset of this work. Investigation should resolve the questions of whether to perform the mapping from syntactic or semantic levels of analysis or combining knowledge from both of them, which of the three forms of UD—basic, enhanced, or parsing—is most suitable for mapping and which conversion approach to adopt (statistical, heuristic).

Another next natural step is to generalize the conversion procedure for other DELPH-IN resource grammars—Japanese Resource Grammar Jacy (Siegel and Bender, 2002), German Resource Grammar GG (Müller and Kasper, 2000), Spanish Resource Grammar SRG (Marimon, 2010), Portuguese Resource Grammar LXGram (Branco and Costa, 2010) and others.

Study of syntactico-semantic dependencies Our contrastive study of syntactico-semantic dependencies from Chapter 3 can be extended by adding more dependency formalisms, such as the aforementioned Universal Dependencies and dependencies from LFG (Cetinoglu et al., 2010), and, similarly to the work of Eskelund (2014), exploring further linguistic characteristics, granularity and variability of formats by computing average depth of dependency trees, combinations of PoS tags and dependency labels, non-projective dependencies and other properties. Apart from Jaccard similarity and unlabeled dependency, there are other symmetric similarity measures appropriate for our task such as inner product, cosine similarity, Pearson correlation and covariance. In order to gauge the utility of various formats, it would be interesting to con-

trast them in such downstream applications as sentiment analysis and semantic role labeling.

Cross-framework parser comparison The obvious extensions of our study on parser comparison are including other PCFG and dependency parsers and trying a wider variety of downstream tasks. The performance of statistical parsers can be improved using tuning and domain adaptation techniques. For example, the set of features for B&N could be enriched with cluster features (e.g. Brown clustering) in order to achieve better domain resilience and the finer-grained ERG lexical categories can be split into several features in order to fight data sparseness while preserving the original information from DT.

Parser combination Parser integration offers a large space for experimentation. There are several venues for future work on this topic. A first obvious step is an in-depth error analysis that would help to better understand the strong and weak sides of parser combination approaches that we have explored in this chapter. In particular, it would be interesting to look at individual dependency types to clarify for which ones there are sufficient gains and losses. Secondly, there is a number of dependency parsers, such as for example ClearParser (Choi and Palmer, 2011) and ZPar (Zhang and Nivre, 2011), that could be used to build various ensembles. Another direction is training a statistical ranker for choosing the high-confidence dependencies instead of building ensembles. We can use unlabeled dependencies as a basis for parser integration, i.e. pruning the search space of PET without dependency labels. For example, Turbo parser has a model for scoring unlabeled dependencies.

Parser combination for improved efficiency of grammar-based parsing can be also tested on other grammars from the DELPH-IN family—Japanese Resource Grammar Jacy (Siegel and Bender, 2002), German Resource Grammar GG (Müller and Kasper, 2000), Spanish Resource Grammar SRG (Marimon, 2010), Portuguese Resource Grammar LXGram (Branco and Costa, 2010) and others. These grammars are compatible with PET. For parser integration experimentation the grammars have to be augmented with an interface for enabling dependency constraints in unification and a procedure for extraction of a dependency backbone from HPSG structures (the latter module is already available for the Spanish Resource Grammar).

An alternative approach to parser combination is enabling the data-driven parser to learn from the grammar-based parser. Preliminary experiments for ERG in this direction have been carried out by Zhang and Wang (2009) who incorporated ERG features into the models of Malt and MST and observed slight performance drops on in-domain data in comparison to the original models and performance gains on the out-of-domain data. For LFG, Øvrelid et al. (2009) realized parser stacking by supplying the data-driven dependency parser with the features derived from the output of the LFG-based parser. The parse accuracy on the enhanced data set significantly surpasses the baseline scores in their experiments. Using features extracted from the PET output to guide a statistical parser, such as B&N, we can test whether grammar-based and data-driven parsers are complementary or overlapping (in case we achieve no performance gains by reparsing).

Bibliography

- Adolphs, Peter, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Bernd Kiefer. Some fine points of hybrid natural language parsing. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May 2008.
- Agić, Željko and Alexander Koller. Potsdam: Semantic Dependency Parsing by Bidirectional Graph-Tree Transformations and Syntactic Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 465–470, Dublin, Ireland, 2014.
- Apresjan, Jurij, Igor Boguslavskij, Leonid Iomdin, Aleksandr Lazurskij, Nikolaj Percov, Vladimir Sannikov, and Leonid Cinman. Lingvističeskoe obespečenie sistemy francuzsko-russkogo avtomatičeskogo perevoda ÈTAP-1 [Linguistic provision of the French-Russian automated machine translation system ETAP-1]. In *Predvaritel'nye pyblikacii Probleмноj gruppy po èksperimental'noj i prikladnoj lingvistike [Preprints of the Problem Group of Experimental and Applied Linguistics]*, volume 154, 155, 159, 160, 166, 167, 174, Moscow, SSSR, 1984-85.
- Apresjan, Jurij, Igor Boguslavskij, Leonid Iomdin, Aleksandr Lazurskij, Nikolaj Percov, Vladimir Sannikov, and Leonid Cinman. *Lingvističeskoe obespečenie sistemy ÈTAP-2 [Linguistic provision of the system ETAP-2]*. Nauka, Moscow, SSSR, 1989.
- Bangalore, Srinivas and Aravind K. Joshi. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999.
- Becker, Tilman and Peter Poller. Two-step TAG parsing revisited. In *In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG4)*, page 143–146, Philadelphia, PA, USA, 1998.
- Bender, Emily M., Dan Flickinger, and Stephan Oepen. The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In Carroll, John, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan, 2002.
- Bender, Emily M., Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. Grammar Customization. *Research on Language & Computation*, 8(1):23–72, 2010. ISSN 1570-7075.

- Bender, Emily M., Dan Flickinger, Stephan Oepen, and Yi Zhang. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, page 397–408, Edinburgh, Scotland, UK, July 2011.
- Bikel, Daniel M. Intricacies of Collins' Parsing Model. *Computational Linguistics*, 30(4): 479–511, 2004.
- Bisk, Yonatan and Julia Hockenmaier. An HDP Model for Inducing Combinatory Categorical Grammars. *Transactions of the Association for Computational Linguistics*, 1:75–87, 2013.
- Black, Ezra, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In *Proceedings of the Workshop on Speech and Natural Language*, pages 134–139, Harriman, New York, USA, 1992.
- Bod, Rens. *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications/Cambridge University Press, Cambridge, US, 1998.
- Bohnet, Bernd. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, page 89–97, Beijing, China, August 2010.
- Bohnet, Bernd and Jonas Kuhn. The best of both worlds: A graph-based completion model for transition-based parsers. In *Proceedings of the 13th Meeting of the European Chapter of the Association for Computational Linguistics*, page 77–87, Avignon, France, 2012.
- Bohnet, Bernd and Joakim Nivre. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, page 1455–1465, Jeju Island, Korea, 2012.
- Booth, Taylor L. Probabilistic representation of formal languages. In *IEEE Conference Record of the 1969 Tenth Annual Symposium on Switching and Automata Theory*, page 74–81, 1969.
- Bos, Johan, Edward Briscoe, Aoife Cahill, John Carroll, Stephen Clark, Ann Copestake, Dan Flickinger, Josef van Genabith, Julia Hockenmaier, Aravind Joshi, Ronald Kaplan, Tracy Holloway King, Sandra Kuebler, Dekang Lin, Jan Tore Lønning, Christopher Manning, Yusuke Miyao, Joakim Nivre, Stephan Oepen, Kenji Sagae, Nianwen Xue, and Yi Zhang, editors. *Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, Manchester, UK, 2008.
- Bosco, Cristina and Alessandro Mazzei. The EVALITA Dependency Parsing Task: From 2007 to 2011. In Magnini, Bernardo, Francesco Cutugno, Mauro Falcone, and Emanuele Pianta, editors, *EVALITA*, volume 7689 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2011.

- Bosco, Cristina, Simonetta Montemagni, Alessandro Mazzei, Vincenzo Lombardo, Felice Dell'Orletta, Alessandro Lenci, Leonardo Lesmo, Giuseppe Attardi, Maria Simi, Alberto Lavelli, Johan Hall, Jens Nilsson, and Joakim Nivre. Comparing the influence of different treebank annotations on dependency parsing. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Valletta, Malta, 2010.
- Bouma, Gosse, Gertjan van Noord, and Rob Malouf. Alpino. Wide-coverage computational analysis of Dutch. In Daelemans, W., K. Sima-an, J. Veenstra, and J. Zavrel, editors, *Computational Linguistics in the Netherlands*, page 45 – 59, Amsterdam, The Netherlands, 2001. Rodopi.
- Branco, Ant3nio and Francisco Costa. A Deep Linguistic Processing Grammar for Portuguese. In *Computational Processing of the Portuguese Language*, volume LNAI6001 of *Lecture Notes in Artificial Intelligence*, page 86 – 89, Berlin, Germany, 2010. Springer.
- Brants, Thorsten. TnT — a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000*, page 224 – 231, Seattle, USA, 2000.
- Bresnan, Joan. *Lexical-Functional Syntax*. Blackwell, Oxford, 2001.
- Briscoe, E. J., J. A. Carroll, and A. Copestake. Relational evaluation schemes. In *Proceedings of the Workshop Beyond Parseval - towards improved evaluation measures for parsing systems at the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, 2002.
- Brown, Peter F., Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based N-gram Models of Natural Language. *Computational Linguistics*, 18(4): 467 – 479, December 1992.
- Buchholz, Sabine and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Natural Language Learning*, page 149 – 164, New York, NY, USA, June 2006.
- Bunt, Harry, Paola Merlo, and Joakim Nivre, editors. *Trends in Parsing Technology. Dependency Parsing, Domain Adaptation, and Deep Parsing*, volume 43. Springer, 2010.
- Burke, Michael, Aoife Cahill, Josef van Genabith, and Andy Way. Evaluating Automatically Acquired F-structures against PropBank. In *Proceedings of the Tenth International Conference on LFG*, pages 84–99, Bergen, Norway, 2005.
- Cahill, Aoife, Mair3ad McCarthy, Josef van Genabith, and Andy Way. Parsing with PCFGs and Automatic F-Structure Annotation. In *Proceedings of the 7th International Conference on LFG*, page 76 – 95, 2002.
- Cahill, Aoife, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. Long-distance Dependency Resolution in Automatically Acquired Wide-coverage PCFG-based LFG Approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 2004.

- Cahill, Aoife, John T. Maxwell, III, Paul Meurer, Christian Rohrer, and Victoria Rosén. Speeding Up LFG Parsing Using C-structure Pruning. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 33–40, Manchester, United Kingdom, 2008.
- Callmeier, Ulrich. PET. A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–108, March 2000.
- Candito, Marie, Joakim Nivre, Pascal Denis, and Enrique Henestroza Anguiano. Benchmarking of statistical dependency parsers for French. In *Proceedings of the 23rd International Conference on Computational Linguistics*, page 108–116, Beijing, China, 2010.
- Carpenter, Bob. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, England, 1992. ISBN 0-521-41932-8.
- Carroll, John and Stephan Oepen. High-efficiency realization for a wide-coverage unification grammar. In Dale, Robert and K. F Wong, editors, *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, volume 3651 of *Lecture Notes in Artificial Intelligence*, page 165–176. Springer, Jeju, Korea, October 2005.
- Carroll, John, Ted Briscoe, and Antonio Sanfilippo. Parser Evaluation: a Survey and a New Proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, page 447–454, Granada, Spain, 1998.
- Carroll, John Andrew. *Practical Unification-based Parsing of Natural Language*. PhD thesis, Computer Laboratory, Cambridge University, 1993.
- Cer, Daniel, Marie-Catherine de Marneffe, Dan Jurafsky, and Chris Manning. Parsing to Stanford Dependencies. Trade-offs between speed and accuracy. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, page 1628–1632, Valletta, Malta, 2010.
- Cetinoglu, Ozlem, Jennifer Foster, Joakim Nivre, Deirdre Hogan, Aoife Cahill, and Josef van Genabith. LFG Without C-Structures. In *Proceedings of the 9th International Workshop on Treebanks and Linguistic Theories*, Tartu, Estonia, 2010.
- Charniak, Eugene. Tree-bank Grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence*, page 1031–1036, Portland, OR, USA, 1996.
- Charniak, Eugene. A maximum-entropy-inspired parser. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, page 132–139, Seattle, WA, USA, April 2000.
- Charniak, Eugene and Mark Johnson. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, page 173–180, Ann Arbor, MI, USA, June 2005.
- Che, Wanxiang, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. A Cascaded Syntactic and Semantic Dependency Parsing System. In *Proceedings of the 12th Conference on Natural Language Learning*, pages 238–242, Manchester, UK, 2008.

- Che, Wanxiang, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. Multilingual Dependency-based Syntactic and Semantic Parsing. In *Proceedings of the 13th Conference on Natural Language Learning*, pages 49–54, Boulder, CO, USA, 2009.
- Choi, Jinho D. and Martha Palmer. Getting the Most out of Transition-based Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 687–692, Portland, Oregon, USA, 2011.
- Chomsky, Noam. *Syntactic Structures*. Mouton, The Hague, 1957.
- Chomsky, Noam. *Aspects of the Theory of Syntax*. Massachusetts Institute of Technology. M.I.T. Press, 1965. ISBN 9780262530071.
- Chomsky, Noam. Remarks on nominalization. In Jacobs, R.A. and P.S. Rosenbaum, editors, *Readings in English transformational grammar*. Ginn, 1970.
- Chomsky, Noam. *Lectures on Government and Binding*. Foris Publications, Dordrecht, 1981.
- Chomsky, Noam. *The Minimalist Program*. Current studies in linguistics series. MIT Press, 1995. ISBN 9780262531283.
- Chu, Y. J. and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14: 1396–1400, 1965.
- Ciaramita, Massimiliano, Giuseppe Attardi, Felice Dell’Orletta, and Mihai Surdeanu. DeSRL: A Linear-time Semantic Role Labeling System. In *Proceedings of the 12th Conference on Natural Language Learning*, pages 258–262, Manchester, UK, 2008.
- Clark, Stephen and James R. Curran. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, page 248–255, Prague, Czech Republic, 2007a.
- Clark, Stephen and James R. Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552, 2007b.
- Clark, Stephen, Julia Hockenmaier, and Mark Steedman. Building Deep Dependency Structures with a Wide-coverage CCG Parser. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, page 327–334, Philadelphia, PA, USA, 2002.
- Coch, José. Interactive generation and knowledge administration in MULTIMETEO. In *Ninth International Workshop on Natural Language Generation*, page 300–303, Niagara-on-the-Lake, Canada, 1998.
- Collins, Michael. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 7th Meeting of the European Chapter of the Association for Computational Linguistics*, page 16–23, Madrid, Spain, July 1997.
- Collins, Michael. *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania, Philadelphia, 1999.

- Collins, Michael. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, volume 10, page 1–8, Philadelphia, PA, USA, 2002.
- Copetake, Ann and Dan Flickinger. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, 2000.
- Copetake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, 3(4):281–332, 2005.
- Corver, N. and H.C. van Riemsdijk. *Semi-lexical Categories: The Function of Content Words and the Content of Function Words*. Studies in generative grammar. Mouton de Gruyter, 2001. ISBN 9783110166859.
- Covington, Michael A. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, page 95–102, Athens, GA, USA, 2001.
- Cramer, Bart and Yi Zhang. Constraining Robust Constructions for Broad-coverage Parsing with Precision Grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 223–231, Beijing, China, 2010.
- Dagan, Ido, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In Quiñonero-Candela, Joaquin, Ido Dagan, Bernardo Magnini, and Florence d’Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, page 177–190. Springer Berlin Heidelberg, 2006.
- de Marneffe, Marie-Catherine and Christopher D. Manning. The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, page 1–8, Manchester, UK, 2008a.
- de Marneffe, Marie-Catherine and Christopher D. Manning. *Stanford typed dependencies manual*. Stanford University, 2008b.
- de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, page 449–454, Genoa, Italy, May 2006.
- de Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies. A cross-linguistic typology. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, page 4585–4592, Reykjavik, Iceland, 2014.
- Devos, Laurent and Michael Gilloux. GPSG Parsing, Bidirectional Charts, and Connection Graphs. In *Proceedings of the 13th International Conference on Computational Linguistics*, volume 2, page 151–155, Helsinki, Finland, 1990.

- Ding, Yuan and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, pages 541–548, Ann Arbor, MI, USA, 2005. Association for Computational Linguistics.
- Dridan, Rebecca. *Using lexical statistics to improve HPSG parsing*. PhD thesis, Department of Computational Linguistics, Saarland University, 2009.
- Dridan, Rebecca. Übertagging. Joint segmentation and supertagging for English. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, page 1–10, Seattle, WA, USA, October 2013a.
- Dridan, Rebecca. Übertagging. Presented at the 9th DELPH-IN Summit, July 2013b. URL <http://www.delph-in.net/2013/bec.pdf>.
- Dridan, Rebecca and Stephan Oepen. Parser evaluation using elementary dependency matching. In *Proceedings of the 12th International Conference on Parsing Technologies*, page 225–230, Dublin, Ireland, October 2011.
- Dridan, Rebecca and Stephan Oepen. Tokenization. Returning to a long solved problem. A survey, contrastive experiment, recommendations, and toolkit. In *Proceedings of the 50th Meeting of the Association for Computational Linguistics*, page 378–382, Jeju, Republic of Korea, July 2012.
- Dridan, Rebecca and Stephan Oepen. Document parsing. Towards realistic syntactic analysis. In *Proceedings of the 13th International Conference on Parsing Technologies*, Nara, Japan, November 2013.
- Edmonds, Jack R. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967.
- Eisner, Jason and Noah Smith. Favor Short Dependencies: Parsing with Soft and Hard Constraints on Dependency Length. In *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, pages 121–150. Springer, 2010.
- Eisner, Jason M. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, page 340–345, Copenhagen, Denmark, 1996.
- Eisner, Jason M. *Smoothing a probabilistic lexicon via syntactic transformations*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2001.
- Elming, Jacob, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Hector Martinez, and Anders Sjøgaard. Down-stream effects of tree-to-dependency conversions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 617–626, Atlanta, Georgia, USA, 2013.
- Eskelund, Norveig. Dependency interconversion. Master thesis, Department of Informatics, University of Oslo, 2014.

- Farkas, Richàrd, Bernd Bohnet, and Helmut Schmid. Features for Phrase-Structure Reranking from Dependency Parses. In *Proceedings of the 12th International Conference on Parsing Technologies*, page 209–214, Dublin, Ireland, 2011.
- Flickinger, Dan. *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford University, Stanford, CA, USA, 1987.
- Flickinger, Dan. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1):15–28, 2000.
- Flickinger, Dan. Toward a Cross-Framework Parser Annotation Standard. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 24–28, Manchester, UK, 2008.
- Flickinger, Dan, Yi Zhang, and Valia Kordoni. DeepBank. A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, page 85–96, Lisbon, Portugal, 2012. Edições Colibri.
- Fokkens, Antske. Metagrammar Engineering: Towards systematic exploration of implemented grammars. In *ACL:11*, page 1066–1076, 2011.
- Foland, William and James Martin. Dependency-Based Semantic Role Labeling using Convolutional Neural Networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288, Denver, Colorado, 2015.
- Foth, A. Kilian and Wolfgang Menzel. Hybrid Parsing: Using Probabilistic Models as Predictors for a Symbolic Parser. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics*, pages 321–328, Sydney, Australia, 2006.
- Fowler, Timothy A. D. and Gerald Penn. Accurate context-free parsing with Combinatory Categorical Grammar. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*, page 335–344, Uppsala, Sweden, 2010.
- Fujita, Sanae, Francis Bond, Stephan Oepen, and Takaaki Tanaka. Exploiting semantic information for hpsg parse selection. In *ACL 2007 Workshop on Deep Linguistic Processing*, page 25–32, Proceedings of the 45th Meeting of the Association for Computational Linguistics, 2007.
- Fundel, Katrin, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.
- Gaifman, Haim. Dependency systems and phrase-structure systems. *Information and Control*, 8(3):304–337, 1965.
- Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford, 1985.

- Gerdes, Kim and Sylvain Kahane. Defining dependencies (and constituents). In *Proceedings of the 1st International Conference on Dependency Linguistics*, page 17–27, Barcelona, Spain, 2011.
- Gildea, Daniel. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, page 167–202, Pittsburgh, USA, 2001.
- Gildea, Daniel and Julia Hockenmaier. Identifying Semantic Roles Using Combinatory Categorical Grammar. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 57–64, Sapporo, Japan, 2003.
- Gildea, Daniel and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28:245–288, 2002.
- Gildea, Daniel and Martha Palmer. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 239–246, Philadelphia, Pennsylvania, 2002.
- Goldberg, Yoav and Michael Elhadad. Inspecting the Structural Biases of Dependency Parsing Algorithms. In *CONLL:10*, pages 234–242, Uppsala, Sweden, 2010.
- Grishman, Ralph, Catherine Macleod, and John Sterling. Evaluating Parsing Strategies Using Standardized Parse Files. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, page 156–161, Trento, Italy, 1992.
- Hajič, Jan. Building a syntactically annotated corpus. The Prague Dependency Treebank. In *Issues of Valency and Meaning*, page 106–132. Karolinum, Prague, Czech Republic, 1998.
- Hajič, Jan, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. Prague Arabic Dependency Treebank: Development in Data and Tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117, Cairo, Egypt, 2004.
- Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th Conference on Natural Language Learning*, page 1–18, Boulder, CO, USA, 2009.
- Hall, Johan. *Transition-Based Natural Language Parsing with Dependency and Constituency Representations*. PhD thesis, Computer Science, Växjö University, 2008.
- Harbusch, Karin. An Efficient Parsing Algorithm for Tree Adjoining Grammars. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics*, page 284–291, Pittsburgh, PA, USA, 1990.

- Harrison, P., S. Abney, E. Black, D. Flickinger, C. Gdaniec, R. Grishman, D. Hindle, B. Ingria, M. Marcus, B. Santorini, and T. Strzalkowski. Evaluating syntax performance of parser/grammars of English. In *Proceedings of the Workshop on Evaluating Natural Language Processing Systems*, page 71 – 77, Berkeley, CA, USA, 1991.
- Hays, David G. Dependency Theory: A Formalism and Some Observations. *Language*, 40(4): 511 – 525, 1964.
- Henderson, John C. and Eric Brill. Exploiting Diversity in Natural Language Processing: Combining Parsers. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, page 187 – 194, College Park, MD, USA, 1999.
- Hockenmaier, Julia and Mark Steedman. CCGbank. A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33:355 – 396, 2007.
- Hopcroft, John E. and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- Horáček, Petr, Eva Zámečnicková, and Ivana Burgetová. Generalized Phrase Structure Grammar. Materials for Lectures FR97/2011/G1 - Formal Models in Natural Language Processing, 2011a. URL <http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:fr97:fmnl04-gpsg.pdf>.
- Horáček, Petr, Eva Zámečnicková, and Ivana Burgetová. Head-Driven Phrase Structure Grammar. Materials for Lectures FR97/2011/G1 - Formal Models in Natural Language Processing, 2011b. URL <http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:fr97:fmnl05-hpsg.pdf>.
- Huddleston, R.D. and G.K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge textbooks in linguistics. Cambridge University Press, 2002. ISBN 9780521431460.
- Hudson, Richard. English word grammar. *Journal of Linguistics*, 28:500–505, 1990.
- Ivanova, Angelina, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. Who Did What to Whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, page 2 – 11, Jeju, Republic of Korea, 2012.
- Ivanova, Angelina, Stephan Oepen, Rebecca Drīdan, Dan Flickinger, and Lilja Øvrelid. On Different Approaches to Syntactic Analysis Into Bi-Lexical Dependencies. An Empirical Comparison of Direct, PCFG-Based, and HPSG-Based Parsers. In *Proceedings of the 13th International Conference on Parsing Technologies*, page 63 – 72, Nara, Japan, 2013a.
- Ivanova, Angelina, Stephan Oepen, and Lilja Øvrelid. Survey on Parsing Three Dependency Representations for English. In *Proceedings of the ACL 2013 Student Research Workshop*, page 31 – 37, Sofia, Bulgaria, 2013b.

- Ivanova, Angelina, Stephan Oepen, Rebecca Dridan, Dan Flickinger, Lilja Øvrelid, and Emanuele Lapponi. On Different Approaches to Syntactic Analysis Into Bi-Lexical Dependencies. An Empirical Comparison of Direct, PCFG-Based, and HPSG-Based Parsers. In press, 2015.
- Johansson, Richard and Pierre Nugues. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, page 105 – 112, Tartu, Estonia, 2007.
- Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136 – 163, 1975. ISSN 0022-0000.
- Judge, John, Aoife Cahill, and Josef van Genabith. QuestionBank: Creating a Corpus of Parse-Annotated Questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics*, page 497 – 504, Sydney, Australia, 2006.
- Jurafsky, Daniel and James H. Martin. *Speech and Language Processing. An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, Upper Saddle River, NJ, USA, 2 edition, 2009.
- Kahane, Sylvain. The Meaning-Text Theory. In Agel, Vilmos, Ludwig M. Eichinger, Hans Werner Eroms, Peter Hellwig, Hans Jurgen Heringer, and Henning Lobin, editors, *Dependency and Valency. An International Handbook on Contemporary Research*, page 546 – 570. de Gruyter, Berlin/New York, 2003.
- Kanerva, Jenna, Juhani Luotolahti, and Filip Ginter. Turku: Broad-Coverage Semantic Parsing with Rich Features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 678–682, Dublin, Ireland, 2014.
- Kaplan, Ronald and Joan Bresnan. Lexical Functional Grammar. A formal system for grammatical representation. In Bresnan, Joan, editor, *The Mental Representation of Grammatical Relations*, page 173 – 281. MIT Press, Cambridge, MA, USA, 1982.
- Kaplan, Ronald M., Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard S. Crouch. Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of Human Language Technology conference (HLT) / North American chapter of the Association for Computational Linguistics annual meeting (NAACL) of the 2004*, pages 97–104, 2004.
- Kawahara, Daisuke, Keiji Shinzato, Tomohide Shibata, and Sadao Kurohashi. Precise Information Retrieval Exploiting Predicate-Argument Structures. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, page 37 – 45, Nagoya, Japan, 2013.
- Kay, Martin. Parsing in Functional Unification Grammar. In Dowty, D. R., L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing*, page 251 – 278. Cambridge University Press, Cambridge, England, 1985.

- Kay, Martin. Readings in Natural Language Processing. chapter Algorithm Schemata and Data Structures in Syntactic Processing, page 35–70. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986.
- Kiefer, Bernd and Hans-Ulrich Krieger. A context-free approximation of Head-driven Phrase Structure Grammar. In *Proceedings of the 6th International Conference on Parsing Technologies*, page 135–146, Trento, Italy, 2000.
- Kiefer, Bernd and Hans-Ulrich Krieger. A Context-Free Superset Approximation of Unification-Based Grammars. In *New Developments in Parsing Technology*. Kluwer, 2004.
- Kiefer, Bernd, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. A Bag of Useful Techniques for Efficient and Robust Parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-ANNUAL'99)*, pages 473–480, College Park, Maryland, USA, 1999.
- Kiefer, Bernd, Hans-Ulrich Krieger, and Detlef Prescher. A Novel Disambiguation Method for Unification-Based Grammars Using Probabilistic Context-Free Approximations. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.
- Kim, Sunghwan Mac, Dominick Ng, Mark Johnson, and James R. Curran. Improving Combinatory Categorical Grammar Parse Reranking with Dependency Grammar Features. In *Proceedings of the 24th International Conference on Computational Linguistics*, page 1441–1458, Mumbai, India, 2012.
- King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, page 1–8, Budapest, Hungary, 2003.
- Kittredge, Richard I. and Alain Polguère. Dependency grammars for bilingual text generation: inside FoG's stratification models. In *Proc. Int. Conf. on Current Issues in Computational Linguistics*, page 318–330, Penang, Malaysia, 1991.
- Klein, Dan and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, page 423–430, Sapporo, Japan, July 2003.
- Klein, Dan and Christopher D. Manning. Corpus-based Induction of Syntactic structure: Models of Dependency and Constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, page 478–485, Barcelona, Spain, 2004.
- Koo, Terry, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology Conference*, page 595–603, Columbus, OH, USA, June 2008.
- Krieger, Hans-Ulrich. From UBGs to CFGs A Practical Corpus-Driven Approach. *Natural Language Engineering*, 13(4):317–351, 2007.

- Krieger, Hans-Ulrich and Ulrich Schäfer. TDL - A Type Description Language for Constraint-Based Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics*, page 893–899, Kyoto, Japan, 1994.
- Kübler, Sandra, Ryan McDonald, Joakim Nivre, and Graeme Hirst. *Dependency Parsing*. Morgan and Claypool Publishers, USA, 2009. ISBN 1598295969.
- Kucera, Henry and W. Nelson Francis. *Computational analysis of present-day American English*. Brown University Press, Providence, RI, 1967.
- Lapponi, Emanuele, Jonathon Read, and Lilja Øvrelid. Representing and resolving negation for sentiment analysis. In *Proceedings of the 2012 ICDM Workshop on Sentiment Elicitation from Natural Text for Information Retrieval and Extraction*, Brussels, Belgium, 2012a.
- Lapponi, Emanuele, Erik Velldal, Lilja Øvrelid, and Jonathon Read. UiO2: sequence-labeling negation using dependency features. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, page 319–327, Montréal, Canada, June 2012b.
- Levin, B. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, 1993. ISBN 9780226475332.
- Lin, Dekang. Dependency-based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, page 317–329, Granada, Spain, 1998.
- Lipton, Zachary Chase, Charles Elkan, and Balakrishnan Narayanaswamy. F1-Optimal Thresholding in the Multi-label Setting. *The Computing Research Repository (CoRR)*, abs/1402.1892, 2014.
- Lønning, Jan Tore, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn NordgÅrd, Victoria Rosén, and Erik Velldal. LOGON. A Norwegian MT effort. Uppsala, Sweden, 2004.
- Ma, Ji, Yue Zhang, and Jingbo Zhu. Punctuation Processing for Projective Dependency Parsing. In *Proceedings of the 52nd Meeting of the Association for Computational Linguistics*, pages 791–796, Baltimore, MD, USA, 2014.
- Magerman, David M. Statistical Decision-tree Models for Parsing. In *Proceedings of the 33th Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts, USA, 1995.
- Malouf, Robert and Gertjan van Noord. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses.*, Hainan Island, China, 2004.
- Manning, Christopher D. Part-of-speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics*, page 171–189, Tokyo, Japan, 2011.

- Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpora of English. The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.
- Marimon, Montserrat. The Spanish Resource Grammar. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, page 700–704, Valletta, Malta, 2010.
- Marimon, Montserrat, Núria Bel, and Lluís Padró. Automatic Selection of HPSG-Parsed Sentences for Treebank Construction. *Computational Linguistics*, 40(3):523–531, 2014.
- Martins, André F. T., Miguel B. Almeida, and Noah A. Smith. Turning on the Turbo: Fast Third-Order Non-Projective Turbo Parsers. In *Proceedings of the 51th Meeting of the Association for Computational Linguistics*, pages 617–622, Sofia, Bulgaria, 2013.
- Martins, T. André F. and C. Mariana S. Almeida. Priberam. A turbo semantic parser with second order features. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, page 471–476, Dublin, Ireland, 2014.
- Matsumoto, Shotaro, Hiroya Takamura, and Manabu Okumura. Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees. In Ho, TuBao, David Cheung, and Huan Liu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3518 of *Lecture Notes in Computer Science*, pages 301–311. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26076-9.
- Matsuzaki, Takuya, Yusuke Miyao, and Jun’ichi Tsujii. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, page 75–82, Ann Arbor, MI, USA, June 2005.
- Matsuzaki, Takuya, Yusuke Miyao, and Jun’ichi Tsujii. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2007)*, page 1671–1676, Hyderabad, India, 2007.
- Maxwell, John T. and Ronald M. Kaplan. The Interface Between Phrasal and Functional Constraints. *Computational Linguistics*, 19(4):571–590, 1993.
- McDonald, Ryan, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, page 91–98, Ann Arbor, MI, USA, 2005a.
- McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, page 523–530, Vancouver, British Columbia, Canada, 2005b.
- McDonald, Ryan, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the 10th Conference on Natural Language Learning*, pages 216–220, 2006.

- McDonald, Ryan T. and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, page 122–131, Prague, Czech Republic, 2007.
- Mehrabi, Saeed, Anand Krishnan, Sunghwan Sohn, Alexandra M. Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul R. Dexter, C. Max Schmidt, Hongfang Liu, and Mathew J. Palakal. DEEPEN: A negation detection system for clinical text incorporating dependency relation into NegEx. *Journal of Biomedical Informatics*, 54:213–219, 2015.
- Mejer, Avihai and Koby Crammer. Confidence in Structured-prediction Using Confidence-weighted Models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 971–981, Cambridge, MA, USA, 2010.
- Mejer, Avihai and Koby Crammer. Are You Sure?: Confidence in Prediction of Dependency Tree Edges. In *Proceedings of Human Language Technologies: The 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 573–576, 2012.
- Mel'čuk, Igor A. *Semantics: From Meaning to Text*, volume 1 of *Semantics: From Meaning to Text*. John Benjamins Publishing Company, 2012. ISBN 9789027205964.
- Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. Annotating noun argument structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, page 803–806, Lisbon, Portugal, 2004.
- Miwa, Makoto, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. Evaluating Dependency Representations for Event Extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 779–787, 2010.
- Miyao, Yusuke and Jun'ichi Tsujii. Probabilistic Disambiguation Models for Wide-Coverage HPSG Parsing. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, page 83–90, Ann Arbor, MI, USA, 2005.
- Miyao, Yusuke and Jun'ichi Tsujii. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80, 2008.
- Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. Corpus-Oriented Grammar Development for Acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, page 684–693, Hainan Island, China, 2004.
- Miyao, Yusuke, Kenji Sagae, and Jun'ichi Tsujii. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the 2007 Workshop on Grammar Engineering across Frameworks*, page 238–258, Palo Alto, California, 2007.

- Miyao, Yusuke, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. Task-oriented Evaluation of Syntactic Parsers and Their Representations. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics*, page 46–54, Columbus, OH, USA, 2008.
- Miyao, Yusuke, Stephan Oepen, and Daniel Zeman. In-House. An ensemble of pre-existing off-the-shelf parsers. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, page 63–72, Dublin, Ireland, 2014.
- Mollá, Diego and Ben Hutchinson. Intrinsic Versus Extrinsic Evaluations of Parsing Systems. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: Are Evaluation Methods, Metrics and Resources Reusable?*, page 43–50, Budapest, Hungary, 2003.
- Montemagni, Simonetta, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, MariaTeresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. Building the Italian Syntactic-Semantic Treebank. In Abeillé, Anne, editor, *Treebanks*, volume 20 of *Text, Speech and Language Technology*, pages 189–210. Springer Netherlands, 2003. ISBN 978-1-4020-1335-5.
- Morante, Roser and Eduardo Blanco. *SEM 2012 Shared Task. Resolving the scope and focus of negation. In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, page 265–274, Montréal, Canada, June 2012.
- Müller, Stefan and Walter Kasper. HPSG analysis of German. In Wahlster, Wolfgang, editor, *VerbMobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer, Berlin, Germany, 2000.
- Nanni, Debbie. On the surface syntax of constructions with *easy*-type adjectives. *Language*, 56 (3):568–591, 1980.
- Neuhaus, Peter and Norbert Bröker. The Complexity of Recognition of Linguistically Adequate Dependency Grammars. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 7th Meeting of the European Chapter of the Association for Computational Linguistics*, page 337–343, Madrid, Spain, 1997.
- Nilsson, Jens. *Transformation and Combination in Data-Driven Dependency Parsing*. PhD thesis, Växjö University, School of Mathematics and Systems Engineering, 2009.
- Ninomiya, Takashi, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. Efficacy of Beam Thresholding, Unification Filtering and Hybrid Parsing in Probabilistic HPSG Parsing. In *Proceedings of the 9th International Conference on Parsing Technologies*, pages 103–114, Vancouver, British Columbia, Canada, 2005.
- Ninomiya, Takashi, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, page 60–68, 2007.

- Nivre, Joakim. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Conference on Parsing Technologies*, page 149–160, Nancy, France, 2003.
- Nivre, Joakim. Dependency grammar and dependency parsing. Technical Report 05133, School of Mathematics and Systems Engineering (MSI), Växjö University, 2005.
- Nivre, Joakim. *Inductive Dependency Parsing*. Text, Speech, and Language Technology. Springer, Dordrecht, The Netherlands, 2006.
- Nivre, Joakim. Sorting Out Dependency Parsing. In Nordstrøm, Bengt and Aarne Ranta, editors, *Advances in Natural Language Processing*, volume 5221 of *Lecture Notes in Computer Science*, pages 16–27. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85286-5.
- Nivre, Joakim. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 1, page 351–359, Suntec, Singapore, 2009.
- Nivre, Joakim. Towards a universal grammar of natural language processing. In *Proceedings of the 16th International Conference on Intelligent Text Processing and Computational Linguistics*, Cairo, Egypt, 2015.
- Nivre, Joakim and Ryan McDonald. Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology Conference*, page 950–958, Columbus, OH, USA, 2008a.
- Nivre, Joakim and Ryan T. McDonald. Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics*, page 950–958, Columbus, OH, USA, 2008b.
- Nivre, Joakim and Jens Nilsson. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics*, page 99–106, Ann Arbor, MI, USA, 2005.
- Nivre, Joakim and Mario Scholz. Deterministic Dependency Parsing of English Text. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 64–70, Geneva, Switzerland, 2004.
- Nivre, Joakim, Johan Hall, and Jens Nilsson. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Natural Language Learning*, page 49–56, Boston, MA, USA, 2004.
- Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, page 915–932, Prague, Czech Republic, June 2007a.

- Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2), 2007b.
- Nivre, Joakim, Laura Rimell, Ryan McDonald, and Carlos Gómez Rodríguez. Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics*, page 833–841, Beijing, China, 2010.
- Oepen, Stephan and John Carroll. Ambiguity packing in constraint-based parsing. Practical results. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, page 162–169, Seattle, WA, USA, 2000.
- Oepen, Stephan and Jan Tore Lønning. Discriminant-based MRS banking. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, page 1250–1255, Genoa, Italy, 2006.
- Oepen, Stephan, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2(4):575–596, 2004.
- Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. SemEval 2014 Task 8. Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, page 63–72, Dublin, Ireland, 2014.
- Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. SemEval 2015 Task 18. Broad-coverage semantic dependency parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, page 915–926, Denver, CO, USA, 2015.
- Oostdijk, Nelleke. The Spoken Dutch Corpus. Overview and first Evaluation. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, 2000.
- Øvrelid, Lilja and Joakim Nivre. When word order and part-of-speech tags are not enough – Swedish dependency parsing with rich linguistic features. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 447–451, 2007.
- Øvrelid, Lilja, Jonas Kuhn, and Kathrin Spreyer. Cross-framework parser stacking for data-driven dependency parsing. *TAL (Traitement Automatique des Langues), special issue on Machine Learning for NLP*, 2009.
- Packard, Woodley. ACE: the Answer Constraint Engine. <http://sweaglesw.org/linguistics/ace/>, 2011. Accessed: 14 August 2015.
- Palmer, Martha, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March 2005. ISSN 0891-2017.

- Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics*, page 433–440, Sydney, Australia, July 2006.
- Plank, Barbara and Gertjan van Noord. Grammar-driven versus data-driven. Which parsing system is more affected by domain shifts? In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground*, page 25–33, Uppsala, Sweden, July 2010. Association for Computational Linguistics.
- Pollard, Carl and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, Chicago, USA, 1994.
- Poon, Hoifung and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2009.
- Popel, Martin and Zdeněk Žabokrtský. TectoMT. Modular NLP framework. *Advances in Natural Language Processing*, page 293–304, 2010.
- Popel, Martin, David Mareček, Jan Štěpánek, Daniel Zeman, and Zdeněk Žabokrtský. Coordination Structures in Dependency Treebanks. In *Proceedings of the 51th Meeting of the Association for Computational Linguistics*, page 517–527, Sofia, Bulgaria, 2013.
- Prins, Robbert and Gertjan van Noord. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139, 2003.
- Pyysalo, Sampo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(1):1–24, 2007.
- Rambow, Owen and Aravind K. Joshi. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. *CoRR*, abs/cmp-lg/9410007, 1994. URL <http://arxiv.org/abs/cmp-lg/9410007>.
- Ramsay, Allan. Effective Parsing with Generalised Phrase Structure Grammar. In *Proceedings of the 2nd Meeting of the European Chapter of the Association for Computational Linguistics*, page 57–61, Geneva, Switzerland, 1985.
- Rasooli, Mohammad Sadegh, Amirsaeid Moloodi, Manouchehr Kouhestani, and Behrouz Minaei Bidgoli. A Syntactic Valency Lexicon for Persian Verbs: The First Steps towards Persian Dependency Treebank. In *5th Language and Technology Conference (LTC)*, page 227–231, Poznań, Poland, 2011.
- Razímová, Magda and Zdeněk Žabokrtský. Morphological Meanings in the Prague Dependency Treebank 2.0. In Matousek, Václav, Pavel Mautner, and Tomáš Pavelka, editors, *Text, Speech and Dialogue*, volume 3658 of *Lecture Notes in Computer Science*, pages 148–155. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28789-6.

- Read, Jonathon, Dan Flickinger, Rebecca Dridan, Stephan Oepen, and Lilja Øvrelid. The We-Search Corpus, Treebank, and Treecache. A comprehensive sample of user-generated content. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, page 1829–1835, Istanbul, Turkey, May 2012.
- Rehbein, Ines and Josef Van Genabith. Evaluating evaluation measures. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, page 372–379, Tartu, Estonia, 2007.
- Ren, Xiaona, Xiao Chen, and Chunyu Kit. Combine Constituent and Dependency Parsing via Reranking. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, page 2155–2161, Beijing, China, 2013.
- Ribeyre, Corentin, Eric Villemonte de la Clergerie, and Djamé Seddah. Alpage: Transition-based Semantic Graph Parsing with Syntactic Features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 97–103, Dublin, Ireland, 2014.
- Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. Parsing the Wall Street Journal Using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 271–278, Philadelphia, PA, USA, 2002.
- Rimell, Laura and Stephen Clark. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865, 2009.
- Rimell, Laura, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, page 813–821, Singapore, 2009.
- Rosenbaum, Peter S. *The grammar of English predicate complement constructions*, volume 46. MIT Press, 1967.
- Sag, Ivan A., Thomas Wasow, and Emily M. Bender. *Syntactic Theory: A Formal Introduction*. CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, California, second edition, 2003. ISBN 9781575863993.
- Sagae, Kenji and Alon Lavie. Parser Combination by Reparsing. In *Proceedings of Human Language Technologies: The 2006 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 129–132, New York City, USA, 2006.
- Sagae, Kenji and Jun’ichi Tsujii. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, page 1044–1050, Prague, Czech Republic, 2007.
- Sagae, Kenji, Yusuke Miyao, and Jun’ichi Tsujii. HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, page 624–631, Prague, Czech Republic, 2007.

- Schmid, Helmut. Efficient Parsing of Highly Ambiguous Context-free Grammars with Bit Vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 162 – 168, Geneva, Switzerland, 2004.
- Schwartz, Roy, Omri Abend, and Ari Rappoport. Learnability-based syntactic annotation design. In *Proceedings of the 24th International Conference on Computational Linguistics*, Mumbai, India, December 2012.
- Sgall, Petr, Eva Hajičová, and Jarmila Panevová. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. D. Reidel Publishing Company, Dordrecht, The Netherlands, 1986.
- Shieber, Stuart M. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes*. CSLI Publications, Stanford, CA, 1986. ISBN 0-937073-00-8.
- Siegel, Melanie and Emily M. Bender. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.
- Snow, Rion, Dan Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics*, Sydney, Australia, 2006.
- Søgaard, Anders, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. What's in a p-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10, Ann Arbor, Michigan, 2014.
- Steedman, Mark. *The Syntactic Process*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0-262-19420-1.
- Straňák, Pavel and Jan Štěpánek. Representing Layered and Structured Data in the CoNLL-ST Format. In *Proceedings of the Second International Conference on Global Interoperability for Language Resources*, page 143 – 152, Hong Kong, China, 2010. City University of Hong Kong. ISBN 978-962-442-323-5.
- Suppes, Patrick, Dan Flickinger, Elizabeth Macken, Jeanette Cook, and Tie Liang. Description of the EPGY Stanford University Online Courses for Mathematics and the Language Arts. In *Proceedings of the International Society for Technology in Education*, San Diego, USA, 2012.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Natural Language Learning*, page 159 – 177, Manchester, UK, 2008.
- Tesnière, Lucien. *Éléments de syntaxe structurale [Elements of Structural Syntax]*. Klincksieck, Paris, 1959.

- Thomson, Sam, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. CMU: Arc-Factored, Discriminative Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 176–180, Dublin, Ireland, 2014.
- Tojo, Satoshi and Ken Saito. Analysis of The Elements by HPSG. In *Proceedings of the 19th Pacific Asia Conference on Language, Information and Computation*, pages 325–332, Taipei, Taiwan, R.O.C., 2005.
- Torisawa, Kentaro, Kenji Nishida, Yusuke Miyao, and Jun-Ichi Tsujii. An HPSG Parser with CFG Filtering. *Natural Language Engineering*, 6(1):63–80, 2000.
- Toutanova, Kristina, Christopher D. Manning, Dan Flickinger, and Stephan Oepen. Stochastic HPSG Parse Disambiguation using the Redwoods Corpus. *Research on Language and Computation*, 3:83–105, 2005.
- Townsend, Richard, Adam Tsakalidis, Yiwei Zhou, Bo Wang, Maria Liakata, Arkaitz Zubiaga, Alexandra Cristea, and Rob Procter. Warwickdcs: From phrase-based to target-specific sentiment recognition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 657–663, Denver, Colorado, 2015.
- Tratz, Stephen and Eduard Hovy. A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, page 1257–1268, Edinburgh, Scotland, UK, 2011.
- Vadas, David and James Curran. Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, page 240–247, Prague, Czech Republic, 2007.
- van der Wouden, Ton, Heleen Hoekstra, Michael Moortgat, Bram Renmans, and Ineke Schuurman. Syntactic Analysis in the Spoken Dutch Corpus (cgn). In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Spain, 2002.
- van Noord, Gertjan. Robust Parsing of Word Graphs. In Junqua, Jean-Claude and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*, volume 17 of *Text, Speech and Language Technology*, pages 205–238. Springer Netherlands, 2001. ISBN 978-90-481-5643-6.
- Čmejrek, Martin, Jan Hajič, and Vladislav Kuboň. Prague Czech-English Dependency Treebank: Syntactically Annotated Resources for Machine Translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004.
- Velldal, Eric. *Empirical Realization Ranking*. PhD thesis, Department of Linguistics, Department of Informatics, University of Oslo, 2009.
- Velldal, Erik, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. Speculation and negation: Rules, rankers and the role of syntax. *Computational Linguistics*, 38(2):369–410, 2012.

- Vilares, David, Miguel A. Alonso, and Carlos Gómez-Rodríguez. A syntactic approach for opinion mining on Spanish reviews. *Natural Language Engineering*, 21:139–163, 1 2015. ISSN 1469-8110.
- Žolkowskij, Aleksandr and Igor Mel'čuk. O vozmožnom metode i instrumentax semantičeskogo sinteza [On potential method and instruments for semantic synthesis]. *Naučno-tehničkaskaja informacija [Scientific and Technological Information]*, page 23–28, 1965.
- Žolkowskij, Aleksandr and Igor Mel'čuk. O semantičeskomo sinteze [On semantic synthesis]. *Problemy kibernetiki [Problems of Cybernetics]*, page 177–238, 1967.
- Wahlster, Wolfgang, editor. *Verbmobil. Foundations of Speech-to-Speech Translation*. Springer, Berlin, Germany, artificial intelligence edition, 2000.
- Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, Prague, Czech Republic, 2007.
- Wetherell, C. S. Probabilistic languages: A review and some open questions. *Computing Surveys*, 12(4):361–379, 1980.
- Wintner, Shuly. Unification Grammars, part II. Materials for Lectures on Computational Linguistics, 2005. URL <http://cs.haifa.ac.il/~shuly/teaching/06/nlp/ug2.pdf>.
- Wu, Shuangzhi, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. Efficient Disfluency Detection with Transition-based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 495–503, Beijing, China, 2015.
- Xia, Fei. *Automatic grammar generation from two different perspectives*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2001.
- Xia, Fei and Martha Palmer. Converting Dependency Structures to Phrase Structures. In *Proceedings of the 1st International Conference on Human Language Technology Research*, page 1–5, San Diego, CA, USA, 2001.
- Yakushiji, Akane, Yusuke Miyao, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. Automatic construction of predicate-argument structure patterns for biomedical information extraction. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 284–292, 2006.
- Yamada, Hiroyasu and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Conference on Parsing Technologies*, page 195–206, Nancy, France, 2003.
- Ytrestøl, Gisle. CuteForce – Deep Deterministic HPSG Parsing. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 186–197, Dublin, Ireland, 2011.

- Ytrestøl, Gisle, Stephan Oepen, and Dan Flickinger. Extracting and annotating Wikipedia sub-domains. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, page 185–197, Groningen, The Netherlands, 2009.
- Zeman, Daniel and Zdeněk Žabokrtský. Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *Proceedings of the 9th International Conference on Parsing Technologies*, page 171–178, Vancouver, British Columbia, Canada, 2005.
- Zhang, Hui, Min Zhang, Chew Lim Tan, and Haizhou Li. K-Best Combination of Syntactic Parsers. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, page 1552–1560, Singapore, 2009a.
- Zhang, Yi and Hans-Ulrich Krieger. Large-scale corpus-driven PCFG approximation of an HPSG. In *Proceedings of the 12th International Conference on Parsing Technologies*, page 198–208, Dublin, Ireland, October 2011.
- Zhang, Yi and Rui Wang. Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, page 378–386, Suntec, Singapore, 2009.
- Zhang, Yi, Valia Kordoni, and Erin Fitzgerald. Partial Parse Selection for Robust Deep Processing. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 128–135, Prague, Czech Republic, 2007a.
- Zhang, Yi, Stephan Oepen, and John Carroll. Efficiency in Unification-Based N-Best Parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, page 48–59, Prague, Czech Republic, July 2007b.
- Zhang, Yi, Rui Wang, and Hans Uszkoreit. Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources. In *Proceedings of the 12th Conference on Natural Language Learning*, page 198–202, Manchester, UK, 2008.
- Zhang, Yi, Rui Wang, and Stephan Oepen. Hybrid Multilingual Parsing with HPSG for SRL. In *Proceedings of the 13th Conference on Natural Language Learning*, pages 31–36, Boulder, CO, USA, 2009b.
- Zhang, Yue and Stephen Clark. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, page 562–571, Honolulu, USA, 2008.
- Zhang, Yue and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technology Conference*, volume 2, page 188–193, Portland, OR, USA, 2011.
- Zhang, Yue and Joakim Nivre. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*, page 1391–1400, Mumbai, India, December 2012.

Zhao, Hai, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. Multilingual Dependency Learning: Exploiting Rich Features for Tagging Syntactic and Semantic Dependencies. In *Proceedings of the 13th Conference on Natural Language Learning*, pages 61–66, Boulder, CO, USA, 2009.

Zhuang, Li, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, page 43–50, Arlington, Virginia, USA, 2006.

Zwicky, Arnold M. Heads. *Journal of Linguistics*, 21:1–29, 3 1985.

