Domain Adaptation for Parsing

Barbara Plank

university of
groningen

faculty of arts

CLCG

The work presented here was carried out under the auspices of the Center for Language and Cognition Groningen (CLCG) at the Faculty of Arts of the University of Groningen and the Netherlands National Graduate School of Linguistics (LOT – *Landelijke Onderzoekschool Taalwetenschap*).

Groningen Dissertations in Linguistics 96
ISSN 0928-0030

Document prepared with LaTeX 2_ε and typeset by pdfTeX.
Cover design and photo by Barbara Plank.
Cover image: *Xanthoria elegans* (commonly known as the elegant sunburst lichen) on a rock in the Dolomites (Alps). The lichen can adapt well to environment changes, it even survived a 16 days exposure to the space (Sancho et al., 2007). Photo taken in the nature park Puez-Geisler on 2,062 m, South Tyrol, Italy, August 5, 2011.
Printed by Wöhrmann Print Service, Zutphen.

RIJKSUNIVERSITEIT GRONINGEN

Domain Adaptation for Parsing

Proefschrift

ter verkrijging van het doctoraat in de
Letteren
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
donderdag 8 december 2011
om 12.45 uur

door

Barbara Plank
geboren op 13 mei 1983
te Bressanone/Brixen, Italië

Promotores:	Prof. dr. G.J.M. van Noord
	Prof. dr. ir. J. Nerbonne

Beoordelingscommissie:	Prof. dr. J. Nivre
	Prof. dr. G. Satta
	Prof. dr. B. Webber

ISBN: 978-90-367-5199-5

Acknowledgments

Just a bit over five years I have now spent here in the Netherlands, and it was a wonderful experience. I would like to thank all people that helped and supported me during this time.

First of all I would like to thank Gertjan van Noord for being an outstanding supervisor and promotor. I am grateful for his guidance, our weekly meetings and especially for the right mix of critical but good advice and freedom to explore my own direction. The fact that he will be wearing his own *toga* at the day of my defense is very well deserved. Moreover, I am grateful to John Nerbonne for being my second promotor. He gave good advice and was always so quick with giving feedback on drafts of this book, which was without doubt one of the crucial ingredients to be able to finish it on time.

Thank goes to the members of my reading committee for their valuable feedback on my dissertation: Joakim Nivre, Giorgio Satta and Bonnie Webber. I am especially grateful to Bonnie Webber for earlier comments on a draft of one of my papers.

I would like to thank Raffaella Bernardi for paving the way with the European Masters Program in Language and Communication Technologies at Bolzano, without which I would not have gotten the taste of this interdisciplinary field of Computational Linguistics and the opportunity to spend a year abroad. Khalil Sima'an supervised my Master thesis at Amsterdam, and I really appreciated to work with him.

After the year in the *randstad* I knew that I would like to continue working in the field. I got the opportunity to extend my stay in the Netherlands. Thanks to Gosse, I landed in the Alfa-informatica hallway, which is such a *gezellige werkplek*. Thanks to my colleagues who are (or were), in a way, part of the group here in Groningen: Çağrı, Daniël, Dicky, Dörte, Erik, Geoffrey, George, Gertjan, Gideon, Giorgos, Gosse, Harm, Hartmut, Henny, Ismail, Jacky, Jelena, Jelle, Johan, John N., John K., Jori, Jörg, Kilian, Kostadin, Leonie, Lonneke, Martijn, Noortje, Nynke, Peter M., Peter N., Proscovia, Sebastian,

Sveta, Tim, Valerio, Wyke, and Yan. Special thank goes to Jelena, Çağrı and Tim, for sharing the office in my first three years and being good friends. Thanks also to Valerio and Dörte for sharing the office in the last year, and Çağrı for sharing it again in my very last month. Moreover, thanks goes to the CLCG sports group for the football afternoons, which I will surely miss.

Furthermore, I would like to thank Sebastian Kürschner for the opportunity to teach a class together. I am grateful to Jörg Tiedemann, who had the initiative to submit a workshop proposal. This gave us the opportunity to organize the Domain Adaptation for Natural Language Processing workshop at ACL 2010. I would like to thank our co-organizers David McClosky, Hal Daumé and Tejaswini Deoskar, the PaCo-MT project for the financial support and John Blitzer for accepting of being our invited speaker. The prior experience of organizing CLIN and TLT 2009 in Groningen was of great help and I would like to thank my fellow Alfa-informatica colleagues for that.

This book additionally benefited from the feedback of many people, some of which I would like to mention here. Marco Wiering from Artificial Intelligence was so kind to check the derivation in the appendix of this dissertation. Daniël read through initial drafts of the parsing chapter, and I am grateful for our discussions on MaxEnt. Çağrı gave valuable feedback on many chapters of this thesis. Jelena commented on the final bits and pieces and gave advice from far away. When it came to designing the cover, I would like to thank Sara for expert tips. Moreover, thanks to our *eekhoorntje*-group (Dörte, Çağrı, Peter) for discussing many of the technical and bureaucratic details about the book and the defense.

I would like to thank my friends in South Tyrol and here in the Netherlands. Thank you Ilona, Erika, Theresia, Pasquale, Barbara, Magda, Andrea, Aska, Gideon M., Magali, and all other friends and relatives. I would like to thank my family, especially my parents and my grandmother. Without their support, especially during the first year in Amsterdam, this adventure would not have been possible. Last but not least, I am grateful to my dear Martin. Thank you for coming to the Netherlands and sharing this experience together, all the support you gave me and your endless love.

Thank you all.
Barbara

Groningen, September 21, 2011

Contents

Chapter 1

Introduction

At last, a computer that understands you like your mother.

— 1985, McDonnell-Douglas ad (L. Lee, 2004)

Natural language processing (NLP) is an exciting research area devoted to the study of computational approaches to human language. The ultimate goal of NLP is to build computer systems that are able to understand and produce natural human language, just as we humans do. Building such systems is a difficult task, given the intrinsic properties of natural language.

One of the major challenges for computational linguistics is *ambiguity* of natural language, exemplified in the very quote above. The quote admits already at least three different interpretations (L. Lee, 2004):

(i) a computer understands you as well as your mother understands you,

(ii) a computer understands that you like your mother,

(iii) a computer understands both you and your mother equally well.

Humans seem to have no problem in identifying the presumably intended interpretation (i), while in general it remains a hard task for a computer. Ambiguity is pertaining to all levels of language; therefore, it is crucial for a practical NLP system to be good at making decisions of, e.g. word sense, word category or syntactic structure (Manning & Schütze, 1999).

In this work, we focus on parsing, the process of syntactic analysis of natural language sentences. A parser is a computational analyzer that assigns syntactic structures (parse structures) to sentences. As such, the ambiguity problem in parsing is characterized by multiple plausible alternative syntactic analyses for a given input sentence. Selecting the most plausible parse tree (or in general, a syntactic structure) is widely regarded as a key to *interpretation* or *meaning*; therefore, the challenge is to incorporate *disambiguation* into processing.

A parser has to choose among the (many) alternative syntactic analyses to find the most likely or plausible parse structure. The framework of probability theory and statistics provides a means to determine the most likely reading for a given sentence and is thus employed as modeling tool, which leads to *probabilistic parsing* (also known as statistical or stochastic parsing).

Domain Dependence of Parsing Systems

Current state-of-the-art statistical parsers employ *supervised machine learning* (ML) to learn (or infer) a model from annotated *training data*. For the task of parsing, the training data consists of a collection of syntactically annotated sentences (a *treebank*).

A fundamental problem in machine learning is that supervised learning systems heavily depend on the data they were trained on. The parameters of a model are estimated to best reflect the characteristics of the training data, at the cost of portability. As a consequence, the performance of such a supervised system drops in an appalling way when the data distribution in the training domain differs considerably from that in the test domain (note that by *domain* we intuitively mean a collection of texts from a certain coherent sort of discourse; however, we will delay a more detailed discussion of the notion of domain until Chapter 3). Thus, a parsing system which is trained on, for instance, newspaper text, will not perform well on data from a different domain, for example, biomedical text. This *problem of domain dependence* is inherent in the assumption of independent and identically distributed (i.i.d.) samples for machine learning (cf. Chapter 3), and thus arises in almost all NLP tasks. However, the problem has started to gain attention only in recent years (e.g. Hara, Miyao & Tsujii, 2005; Daumé III, 2007; McClosky, Charniak & Johnson, 2006; Jiang & Zhai, 2007).

One possible approach to solving this problem is to annotate data from the new domain. However, annotating data is expensive and is therefore unsatisfactory. Therefore, the goal of *domain adaptation* is to develop algorithms that allow the adaptation of NLP systems to new domains without incurring the undesirable costs of annotating new data.

The focus of this dissertation is on domain adaptation for natural language parsing systems. More specifically, after setting the theoretical background of this work (part I), in part II we will investigate adaptation approaches for the syntactic disambiguation component of a grammar-driven parser. While most previous work on domain adaptation has focused on data-driven parsing systems, we will investigate domain adaptation techniques for the syntactic disambiguation component of Alpino, a grammar-driven dependency parser for Dutch (cf. Chapter 2 for a definition of data-driven and grammar-driven parsing systems). The research question that will be addressed in part II of this dissertation is the following:

Q1 How effective are domain adaptation techniques in adapting the syntactic disambiguation model of a grammar-driven parser to new domains?

We will examine techniques that assume a limited amount of labeled data for the new domain as well as techniques that require only unlabeled data.

Then, in part III we extend our view to multiple parsing systems and compare the grammar-driven system to two data-driven parsers to find an answer to the following question:

Q2 Grammar-driven versus data-driven: Which parsing system is more affected by domain shifts?

We investigate this issue to test our hypothesis that the grammar-driven system is less affected by domain shifts, and consequently, data-driven systems are more in need for domain adaptation techniques.

As we discuss in Chapter 3, most previous work on domain adaptation relied on the assumption that there is (labeled or unlabeled) data available for the new target domain. However, with the increasing amounts of data that become available, a related yet rather unexplored issue arises, that we will investigate in part IV:

Q3 Given training data from several source domains, what data should we use to train a parser for a new target domain, i.e. which similarity measure is good for parsing?

In order to answer this question, we need a way to measure the similarity between the domains. Therefore, the last chapter focuses on evaluating several measures of domain similarity to gather related training data for a new unknown target domain. An empirical evaluation on Dutch and English is carried out to adapt a data-driven parsing system to new domains. The following section provides a more detailed outline of this dissertation.

Chapter Guide

Chapter 2 describes the task of parsing and its challenges and introduces two major grammar formalisms with their respective probability models. The chapter also provides an overview of the parsing systems used in this work. They include Alpino, a grammar-driven parsing system for Dutch that employs a statistical parse selection component (also known as parse disambiguation component), and two data-driven dependency parsers, MST and Malt.

Chapter 3 introduces the problem of the domain dependence of natural language processing systems, which is a general problem of supervised machine learning. The chapter provides an overview of the field with an emphasis on the task of parsing, and introduces straightforward baselines as well as specific domain adaptation techniques. The chapter also discusses the notion of *domain* and how it was perceived in previous work.

Chapter 4 and Chapter 5 focus on applying domain adaptation techniques to adapt the statistical parse selection component of the Alpino parser to new domains. Chapter 4 examines the scenario in which there is some limited amount of labeled data available for the new target domain (the *supervised domain adaptation* setting). In contrast, Chapter 5 explores techniques for the case when only unlabeled data is available (*unsupervised domain adaptation*).

Chapter 6 presents an empirical investigation of the problem of domain sensitivity of different parsing systems. While the focus of the previous two chapters is solely on the disambiguation component of the Alpino parser, this chapter analyzes the behavior of different types of parsing systems when facing a domain shift. The hypothesis tested is that the grammar-driven system Alpino is less affected by domain shifts, in comparison to purely data-driven statistical parsing systems, such as MST and Malt. The chapter presents the results of an empirical investigation on Dutch.

Chapter 7 presents an effective way to measure domain similarity. Most previous work on domain adaptation assumed that domains are given (i.e. that they are represented by the respective corpora). Thus, one knew the target domain, had some labeled or unlabeled data of that domain at disposal, and tried to adapt the system from one domain to another. However, as more data becomes available it is less likely that domains will be given. Thus, automatic ways to select data to train a model for a target domain are becoming attractive. The chapter shows a simple and effective way to automatically measure domain similarity to select the most similar data for a new test set.

The final chapter summarizes and concludes this thesis, discusses limitations of proposed approaches and provides directions for future research.

Publications

Parts of this dissertation are based on (or might refer to) the following publications. Footnotes at the beginning of the chapters indicate which publication(s) is/are relevant for the respective chapter.

Plank, B. & van Noord, G. (2008). Exploring an Auxiliary Distribution Based approach to Domain Adaptation of a Syntactic Disambiguation Model. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation* (pp. 9–16). Manchester, UK.

Plank, B. & Sima'an, K. (2008). Subdomain Sensitive Statistical Parsing using Raw Corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. Marrakech, Morocco.

Plank, B. (2009b). Structural Correspondence Learning for Parse Disambiguation. In *Proceedings of the Student Research Workshop at EACL 2009* (pp. 37–45). Athens, Greece.

Plank, B. (2009a). A Comparison of Structural Correspondence Learning and Self-training for Discriminative Parse Selection. In *Proceedings of the NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing* (pp. 37–42). Boulder, Colorado, USA.

Plank, B. & van Noord, G. (2010a). Dutch Dependency Parser Performance Across Domains. In E. Westerhout, T. Markus and P. Monachesi (Eds.), *Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands* (pp. 123-138). Utrecht, The Netherlands.

Plank, B. & van Noord, G. (2010b). Grammar-driven versus Data-driven: Which Parsing System is More Affected by Domain Shifts? In *Proceedings of the ACL Workshop on NLP and Linguistics: Finding the Common Ground* (pp. 25–33). Uppsala, Sweden.

Plank, B. & van Noord, G. (2011). Effective Measures of Domain Similarity for Parsing. In *Proceedings of the 49th Meeting of the Association for Computational Linguistics* (pp. 1566-1576). Portland, Oregon, USA.

Part I

Background

Chapter 2

Natural Language Parsing

In this chapter, we first define the task of parsing and its challenges. We will give a conceptual view of a parsing system and discuss possible instantiations thereof. Subsequently, we will introduce two major grammar formalisms with their corresponding probability models. Finally, we will give details of the parsing systems and corpora used in this work. A large part of the chapter will be devoted to Alpino, a grammar-driven parser for Dutch, because its parse selection component is the main focus of the domain adaptation techniques explored in Chapter 4 and Chapter 5. The chapter will end with a somewhat shorter introduction to two data-driven dependency parsing systems (MST and Malt), that will be used in later chapters of this thesis.

2.1 Parsing

Parsing is the task of identifying the syntactic structure of natural language sentences. The syntactic structure of a sentence is key towards identifying its meaning; therefore, parsing is an essential task in many natural language processing (NLP) applications.

However, natural language sentences are often ambiguous, sometimes to an unexpectedly high degree. That is, for a given input there are multiple alternative linguistic structures that can be built for it (Jurafsky & Martin, 2008). For example, consider the sentence:

(2.1) Betty gave her cat food

Two possible syntactic structures for this sentence are given in Figure 2.1 (these are *phrase structure* trees). The leaves of the trees (terminals) are the words together with their part-of-speech (PoS) tags, e.g. PRP is a personal pronoun, PRP$ is a possessive pronoun.[1] The upper nodes in the tree are non-terminals and represent larger phrases (constituents), e.g. the verb phrase VP 'gave her cat food'. The left parse tree in Figure 2.1 represents the meaning of Betty giving food to her cat. The right parse tree stands for Betty giving cat food to 'her', which could be another female person. A more compact but equivalent representation of a phrase-structure tree is the *bracketed notation*. For instance, the left parse tree of Figure 2.1 can be represented in bracketed notation as: [S [NP [NNP Betty]] [VP [VDB gave] [NP [PRP$ her] [NN cat]] [NN food]]].

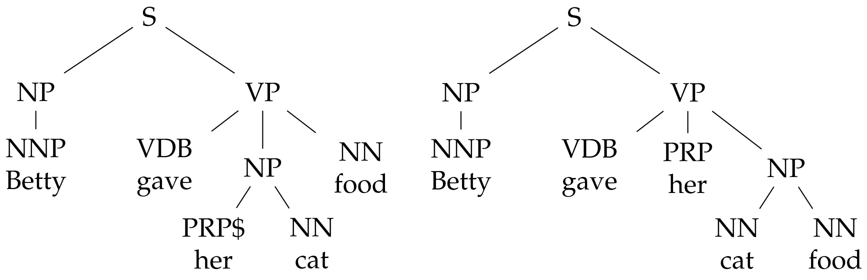


Figure 2.1: Two phrase structures for sentence (2.1).

In other formalisms, the structure of a sentence is represented as *dependency structure* (also known as *dependency graph*). An example is shown in Figure 2.2 (note that sometimes the arcs might be drawn in the opposite direction). Instead of focusing on constituents and phrase-structure rules (as in the phrase-structure tree before), the structure of a sentence is described in terms of binary relations between words, where the syntactically subordinate word is called the dependent, and the word on which it depends is its head (Kübler, McDonald & Nivre, 2009). The links (*edges* or *arcs*) between words are called *dependency relations* and usually indicate the type of dependency relation. For instance, 'Betty' is the subject (sbj) dependent of the head-word 'gave'.[2]

Humans usually have no trouble in identifying the intended meaning (e.g. the left structures in our example), while it is a hard task for a natural language processing system. *Ambiguity* is a problem pertaining to all levels of natural language. The example above exemplifies two kinds of ambiguity:

[1] These are the Penn Treebank tags. A description of them can be found in Santorini (1990).

[2] A peculiarity of the structure in Figure 2.2 is that an artificial ROOT token has been added. This ensures that every word in the sentence has (at least) one associated head-word.

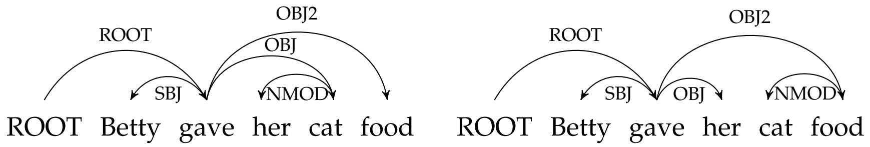


Figure 2.2: Two dependency graphs for sentence (2.1) (PoS tags omitted).

structural ambiguity (if there are multiple alternative syntactic structures) and *lexical ambiguity* (also called *word-level ambiguity*, e.g. whether 'her' is a personal or possessive pronoun). Thus, the challenge in parsing is to incorporate *disambiguation* to select a single preferred reading for a given sentence.

Conceptually, a parsing system can be seen as a two-part system, as illustrated in Figure 2.3. The first part is the *parsing component*, a device that employs a mechanism (and often further information in the form of a *grammar*) to generate a set of possible syntactic analyses for a given input (a sentence). The second part is the *disambiguation component* (also known as *parse selection* component), that selects a single preferred syntactic structure. Hence, the job of a parser consists, besides finding the syntactic structure, also in deciding which parse to choose in case of ambiguity.

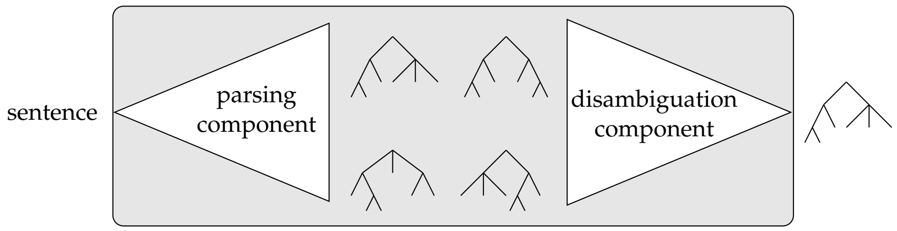


Figure 2.3: A parsing system - conceptual view (inspired by lecture notes of Khalil Sima'an, University of Amsterdam).

The framework of probability theory and statistics provides a means to determine the plausibility of different parses for a given sentence and is thus employed as modeling tool, which leads to *statistical parsing* (also known as probabilistic or stochastic parsing). Statistical parsing is the task of finding the most-plausible parse tree y for a given sentence x according to a model M that assigns a score to each parse tree $y \in \Omega(x)$, where $\Omega(x)$ is the set of possible parse trees of sentence x:

$$y^* = \arg\max_{y \in \Omega(x)} score(x, y) \tag{2.2}$$

To compute the score of the different parses for a given sentence we need a model M. Therefore, one has to complete two tasks: (i) define how the score of a parse is computed, i.e. define the structure of the model; (ii) instantiate the model, which is the task of training or parameter estimation.

One way to model the score of a parse is to consider the joint probability $p(x, y)$. This follows from the definition of conditional probability $p(y|x) = p(x, y)/p(x)$ and two observations: In parsing, the string x is already given, and is implicit in the tree (i.e. its yield). Therefore, $p(x)$ is constant and can be effectively ignored in the maximization. *Generative models* estimate $p(x, y)$ and thus define a model over all possible (x, y) pairs. The underlying assumption of generative parsing models is that there is a stochastic process that generates the tree through a sequence of steps (a derivation), so that the probability of the entire tree can be expressed by the product of the probabilities of its parts. This is essentially the decomposition used in probabilistic context-free grammars (PCFGs), to which we will return to in the next section.

In contrast, models that estimate the conditional distribution $p(y|x)$ directly (rather than indirectly via the joint distribution) are called *discriminative models*. Discriminative parsing models have two advantages over generative parsing models (Clark, 2010): they do not spend modeling efforts on the sentence (which is given anyway in parsing) and it is easier to incorporate complex features into such a model. This is because discriminative models do not make explicit independence assumptions in the way that generative models do. However, the estimation of model parameters becomes harder, because simple but efficient estimators like empirical relative frequency fail to provide a consistent estimator (Abney, 1997), as will be discussed further later. Note, however, that there are statistical parser that do not explicitly use a probabilistic model. Rather, all that is required is a ranking function that calculates scores for the alternative readings.

Before moving on, we will discuss various instantiations of the conceptual parsing schema given in Figure 2.3. As also proposed by Carroll (2000), we can divide parsing systems into two broad types: *grammar-driven* and *data-driven* systems. Note that the boundary between them is somewhat fuzzy, and this is not the only division possible. However, it characterizes nicely the two kinds of parsing systems we will use in this work.

- *Grammar-driven systems:* These are systems that employ a formal (often hand-crafted) grammar to generate the set of possible analyses for a given input sentence. There is a separate second-stage: a statistical disambiguation component that selects a single preferred analysis. Training such a system means estimating parameters for the disambiguation component only, as the grammar is given. Examples of such systems are: Alpino, a parser for Dutch – it is used in this work and will be introduced in Section 2.4.1; PET, a parser that can use grammars of various languages, for instance the English resource grammar (Flickinger, 2000).
- *Data-driven systems:* Parsing systems that belong to this category automatically induce their model or grammar from an annotated corpus (a *treebank*). Examples of such parsing systems are data-driven dependency parsers, such as the MST (McDonald, Pereira, Ribarov & Hajič, 2005) and Malt (Nivre et al., 2007) parsers. They will be introduced in Section 2.4.2 and are used in later chapters of this thesis.

To some extent, these two approaches can be seen complementary, as there are parsing systems that combine elements of both approaches. For instance, probabilistic context-free grammars (Collins, 1997; Charniak, 1997) (they are discussed in further detail in Section 2.2) are both grammar-based and data-driven. While Carroll (2000) considers them as grammar-driven systems, we actually find them somewhat closer to data-driven systems. They do employ a formal grammar – however, this grammar is usually automatically acquired (induced) from a treebank. Moreover, PCFGs generally integrate disambiguation directly into the parsing stage. However, there exist systems that extend a standard PCFG by including a separate statistical disambiguation component (also called a reranker) that reorders the n-best list of parses generated by the first stage (Charniak & Johnson, 2005).

In the following, we will discuss two well-known grammar formalisms and their associated probability models: probabilistic context-free grammars (PCFGs) and attribute-value grammars (AVGs). The chapter will end with a description of the different parsing systems used in this work.

2.2 Probabilistic Context-Free Grammars

The most straightforward way to build a statistical parsing system is to use a *probabilistic context-free grammar* (PCFG), also known as stochastic context-free grammar or phrase-structure grammar. It is a grammar formalism that underlies many current statistical parsing systems today (e.g. Collins, 1997; Charniak, 1997; Charniak & Johnson, 2005). A PCFG is the probabilistic version of a context-free grammar.

A *context-free grammar* (CFG) is a quadruple $(V_N, V_T, S, \mathcal{R})$, where V_N is the finite set of non-terminal symbols, V_T is the finite set of terminal symbols (lexical elements), S is a designated start symbol and $\mathcal{R}$ is a finite set of rules of the form $r_i : V_N \rightarrow \zeta$, where $\zeta = (V_T \cup V_N)^*$ (a sequence of terminals and nonterminals). The rules are also called production or phrase-structure rules. A CFG can be seen as a rewrite rule system: each application of a grammar rule rewrites its left-hand side with the sequence ζ on the right-hand side. By starting from the start symbol S and applying rules of the grammar, one can derive the parse tree structure for a given sentence (cf. Figure 2.5). Note that several derivations can lead to the same final parse structure.

A *probabilistic context-free grammar* (PCFG) extends a context-free grammar by attaching a probability $p(r_i) \in [0, \dots, 1]$ to every rule in the grammar $r_i \in \mathcal{R}$. The probabilities are defined such that the probabilities of all rules with the same antecedent $A \in V_N$ sum up to one: $\forall A : \sum_j p(A \rightarrow \zeta^j) = 1$. That is, there is a probability distribution for all possible daughters for a given head.[3] An example PCFG, taken from Abney (1997), is shown in Figure 2.4.

	$r_i \in \mathcal{R}$	$p(r_i)$		$r_i \in \mathcal{R}$	$p(r_i)$
(i)	$S \rightarrow AA$	1/2	(iv)	$A \rightarrow b$	1/3
(ii)	$S \rightarrow B$	1/2	(v)	$B \rightarrow aa$	1/2
(iii)	$A \rightarrow a$	2/3	(vi)	$B \rightarrow bb$	1/2

Figure 2.4: Example PCFG: $V_N = \{A, B\}, V_T = \{a, b\}$ and set of rules $\mathcal{R}$ with associated probabilities.

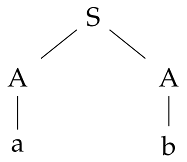
That is, in a PCFG there is a proper probability distribution over all possible expansions of any non-terminal. In such a model it is assumed that there is a generative process that builds the parse tree in a Markovian fashion: elements are combined, where the next element only depends on the former element in the derivation process (the left-hand side non-terminal). Thus, the expansion of a non-terminal is independent of the context, that is, of other elements in the parse tree. Based on this independence assumption, the probability of a parse tree is simple calculated as the product of the probabilities of all rule applications used in building the tree $r_i \in \mathcal{R}(y)$. More formally, let

[3]Thus, like Manning and Schütze (1999, chapter 11), when we write $\forall A : \sum_j p(A \rightarrow \zeta^j) = 1$ we actually mean $\forall A : \sum_j p(A \rightarrow \zeta^j | A) = 1$.

$c(r_i)$ define the count of how often rule r_i has been used in the derivation of tree y, then:

$$p(x,y) = \prod_{r_i \in \mathcal{R}(y)} p(r_i)^{c(r_i)}$$

For example, given the PCFG above, Figure 2.5 illustrates the derivation of a parse tree and its associated probability calculation. To find the most likely parse for a sentence, a widely-used dynamic programming algorithm is the CKY (Cocke-Kasami-Younger) chart parsing algorithm, described in detail in e.g. Jurafsky and Martin (2008, chapter 14).



Rules used in derivation:
(i) $S \rightarrow AA$, (iii) $A \rightarrow a$, (iv) $A \rightarrow b$

$$\begin{aligned} p(x,y) &= p(S \rightarrow AA) \times p(A \rightarrow a) \times p(A \rightarrow b) \\ &= 1/2 \times 2/3 \times 1/3 = 1/9 \end{aligned}$$

Figure 2.5: Example derivation for the PCFG given in Figure 2.4.

If we have access to a corpus of syntactically annotated sentences (a *treebank*), then "the simplest way to 'learn' a context-free grammar [...] is to read the grammar off the parsed sentences" (Charniak, 1996). The grammar acquired in this way is therefore also called *treebank grammar* (Charniak, 1996). The first step is to extract rules from the treebank by decomposing the trees that appear in the corpus. The second step is to estimate the probabilities of the rules. For PCFGs this can be done by relative frequency estimation (Charniak, 1996), since the relative frequency estimator provides a maximum likelihood estimator in the case of PCFGs (Abney, 1997):

$$p(\alpha \rightarrow \zeta) = \frac{\text{count}(\alpha \rightarrow \zeta)}{\text{count}(\alpha)} \tag{2.3}$$

However, despite their simplicity and nice theoretical properties, PCFGs have weaknesses (Jurafsky & Martin, 2008; Manning & Schütze, 1999). The two main problems of standard PCFGs are: (i) the lack of sensitivity to structural preferences; (ii) the lack of sensitivity to lexical information. These problems all stem from the independence assumptions made by PCFGs. Recall that the application of a rule in a PCFG is independent of the context – it is conditioned only on the previous (parent) node. As such, a PCFG does not capture important lexical and structural dependencies.

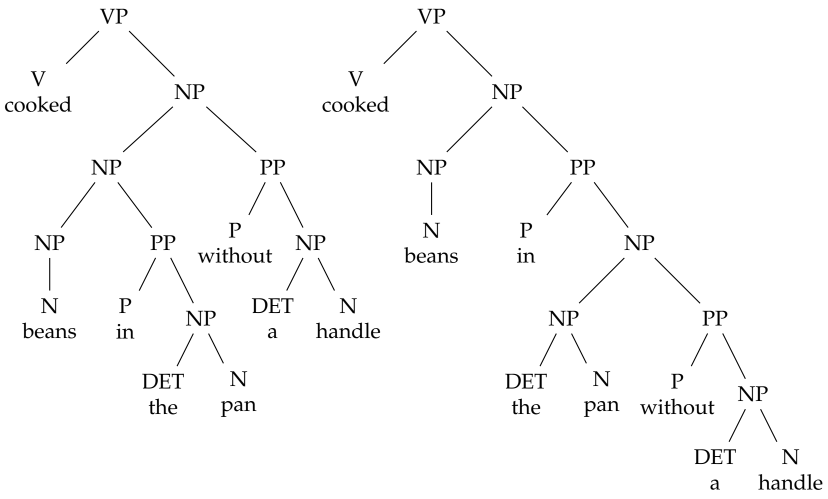


Figure 2.6: An instance of a PP (prepositional phrase) ambiguity. Although the right structure is the more likely one (the pan has no handle, not the beans), a PCFG will assign equal probability to these competing parses since both use exactly the same rules.

For instance, let's consider subject-verb agreement. A grammar rule of the form `S→NP VP` does not capture agreement, since it does not prevent that the NP is rewritten e.g. into a plural noun (e.g. 'cats') while the VP expands to a singular verb (e.g. 'meows'), thus giving 'cats meows'*. Another example, taken from Collins (1999, ch.3), is attachment: 'workers dumped sacks into a bin'. Two possible parse trees for the sentence are (in simplified bracketed notation): (a) 'workers [dumped sacks] [into a bin]'; (b) 'workers [dumped sacks [into a bin]]. That is, they differ in whether the prepositional phrase (PP) 'into a bin' attaches to the verb phrase (VP) 'dumped sacks' as in (a), or instead attaches to the noun phrase 'sacks' as in (b). Thus, the two parse trees differ only by one rule (either `VP→VP PP` or `NP→NP PP`). That is, the probabilities of these rules alone determine the disambiguation of the attachment – there is no dependence on the lexical items themselves (Collins, 1999). Figure 2.6 shows an even more extreme example: the PCFG does not encode any preference over one of the two possible structures because exactly the same rules are used in the derivation of the trees.

To overcome such weaknesses, various extensions of PCFGs have been introduced, for instance, lexicalized PCFGs (Collins, 1997). They incorporate lexical preferences by transforming a phrase structure tree into a *head-lexicalized parse tree* by associating to every non-terminal in the tree its head word. An example is illustrated in Figure 2.7. Another mechanism is parent annotation, proposed by Johnson (1998), where every non-terminal node is associated with its parent. We will not discuss these extensions here further, as PCFGs are not used in this work. Rather, we will move now to a more powerful grammar formalisms, namely, attribute-value grammars.

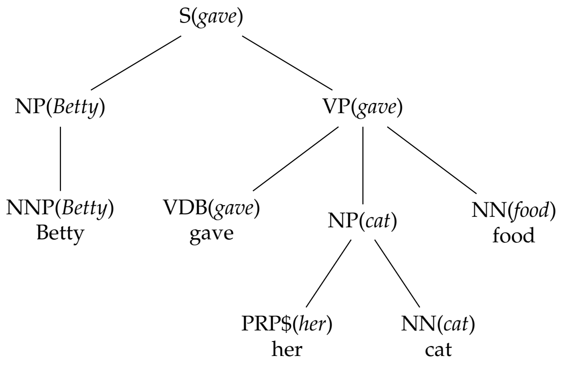


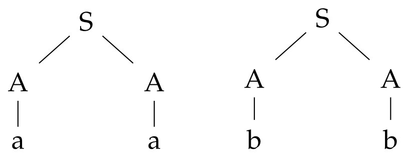
Figure 2.7: A lexicalized phrase structure tree (Collins, 1997).

2.3 Attribute-Value Grammars and Maximum Entropy

Attribute-value grammars are an extension of context-free grammars (CFGs). Context-free grammars provide the basis of many parsing systems today, despite their well-known restrictions in capturing certain linguistic phenomena, such as agreement and attachment (discussed above), or other linguistic phenomena like coordination (e.g. 'dogs in houses and cats', taken from Collins (1999), i.e. whether dogs and cats are coordinated, or houses and cats), long-distance dependencies, such as wh-relative clauses ('This is the player who the coach praised.') or topicalization ('On Tuesday, I'd like to fly from Detroit to Saint Petersburg').

2.3.1 Attribute-Value Grammars

Attribute-value grammars (AVGs) extend context-free grammars (CFGs) by adding constraints to the grammar. Therefore, such grammar formalisms are also known as constraint-based grammars.



S → A A
A → a
A → b

Figure 2.8: Example treebank and induced CFG.

For instance, consider the treebank given in Figure 2.8, taken from Abney (1997). The treebank-induced context-free grammar would not capture the fact the two non-terminals should rewrite to the same symbol. Such *context dependencies* can be imposed by means of attribute-value grammars (Abney, 1997). An attribute-value grammar can be formalized as a CFG with attribute labels and path equations. For example, to impose the constraint that both non-terminals *A* rewrite to the same orthographic symbol, the grammar rules are extended as shown in Figure 2.9 (i.e. the ORTH arguments need to be the same for both non-terminals). The structures resulting from AV grammars are directed acyclic graphs (*dags*), and no longer only trees, as nodes might be shared.

$$\begin{array}{rl} S \rightarrow & AA \\ & \langle A \text{ ORTH} \rangle = \langle A \text{ ORTH} \rangle \\ A \rightarrow & a \\ & \langle A \text{ ORTH} \rangle = a \\ A \rightarrow & b \\ & \langle A \text{ ORTH} \rangle = b \end{array}$$

Figure 2.9: Augmented grammar rules including constraints on the orthographic realizations of the non-terminals. This grammar generates the trees shown in Figure 2.8, while it correctly fails to generate a parse tree where both non-terminals would rewrite to different terminal symbols, i.e. *ab*.

In more detail, in such a formalism atomic categories are replaced with complex feature structures to impose constraints on linguistic objects. These feature structures are also called attribute-value structures. They are more commonly represented as attribute-value matrices (AVMs). An AVM is a list of attribute-value pairs. A value of a feature can be an atomic value or another feature structure (cf. Figure 2.10). To specify that a feature value is shared (also known as reentrant), coindexing is used (Shieber, 1986). For instance, Figure 2.11 shows that the verb and its subject share the same (number and person) agreement structure.

$$\begin{bmatrix} \mathit{CAT} & \mathit{A} \\ \mathit{ORTH} & \mathit{a} \end{bmatrix} \qquad\qquad \begin{bmatrix} \mathit{CAT} & \mathit{NP} \\ \mathit{AGR} & \begin{bmatrix} \mathit{NUM} & \mathit{singular} \\ \mathit{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

Figure 2.10: Feature structures with atomic (left) and complex (right) feature values.

$$\begin{bmatrix} \mathit{CAT} & \mathit{VP} \\ \mathit{AGR} & \mathbf{1}\begin{bmatrix} \mathit{NUM} & \mathit{singular} \\ \mathit{PERS} & 3 \end{bmatrix} \\ \mathit{SBJ} & \begin{bmatrix} \mathit{AGR} & \mathbf{1} \end{bmatrix} \end{bmatrix}$$

Figure 2.11: Feature structure with reentrancy.

To combine feature structures a mechanism called unification is employed. It ensures that only compatible feature structures combine into a new feature structure. Therefore, attribute-value grammars are also known as unification-based grammars. Grammars that are based on attribute-value structures and unification include formalisms such as the lexical functional grammar (LFG) and head-driven phrase structure grammar (HPSG).

However, the property of capturing context-sensitivity comes at a price: stochastic versions of attribute-value grammars are not that simple as in the case of PCFGs. As shown by Abney (1997), the straightforward relative frequency estimate (used by PCFGs) is not appropriate for AVGs. It fails to provide a maximum likelihood estimate in the case of dependencies found in attribute-value grammars. Therefore, a more complex probability model is needed. One solution is provided by maximum entropy models. The Alpino parser (Section 2.4.1) is a computational analyzer for Dutch based on a HPSG-like grammar formalism. It uses a hand-crafted attribute-value grammar with a large lexicon and employs the maximum entropy framework for disambiguation, introduced next.

2.3.2 Maximum Entropy Models

Maximum entropy (or short: MaxEnt) models provide a general-purpose machine learning (ML) method that has been widely used in natural language processing, for instance in PoS tagging (Ratnaparkhi, 1998), parsing (Abney, 1997; Johnson, Geman, Canon, Chi & Riezler, 1999) and machine translation (Berger, Della Pietra & Della Pietra, 1996).

A maximum entropy model is specified by a set of features f_i and their associated weights λ_i. The features describe properties of the data instances (*events*). For example in parsing, an event might be a particular sentence-parse pair and a feature might describe how often a particular grammar rule has been applied in the derivation of the parse tree. During training, feature weights are estimated from training data. The maximum entropy principle provides a guideline to choose one model out of the many models that are consistent with the training data.

In more detail, a training corpus is divided into observational units called events (e.g. sentence-parse pairs). Each event is described by a m-dimensional real-valued feature vector function f:

$$\forall i \in [1, \ldots, m] : f_i(x, y) \rightarrow \mathbb{R} \tag{2.4}$$

The feature function f maps a data instance (x, y) to a vector of $\mathbb{R}$-valued feature values. Thus, a training corpus represents a set of statistics that are considered useful for the task at hand.

During the training procedure, a model p is constructed that satisfies the constraints imposed by the training data. In more detail, the expected value of a feature f_i of the model p to be learned, $E_p[f_i]$:

$$E_p[f_i] = \sum_{x,y} p(x, y) f_i(x, y) \tag{2.5}$$

has to be equal to $E_{\tilde{p}}[f_i]$, the expected value of feature f_i as given by the empirical distribution $\tilde{p}$ obtained from the training data:

$$E_{\tilde{p}}[f_i] = \sum_{x,y} \tilde{p}(x, y) f_i(x, y) \tag{2.6}$$

That is, we require the model to constrain the expected value to be the same as the expected value of the feature in the training sample:

$$\forall i, E_p[f_i] = E_{\tilde{p}}[f_i] \tag{2.7}$$

or, more explicitly:

$$\forall i, \sum_{x,y} p(x,y) f_i(x,y) = \sum_{x,y} \tilde{p}(x,y) f_i(x,y) \tag{2.8}$$

In general, there will be many probability distributions that satisfy the constraints posed in equation (2.8). The *principle of maximum entropy* argues that the best probability distribution is the one which maximizes entropy, because:

> ... it is the least biased estimate possible on the given information; i.e., it is maximally noncommittal with regard to missing information. [...] to use any other [estimate] would amount to arbitrary assumption of information which by hypothesis we do not have. (Jaynes, 1957).

Among all models $p \in P$ that satisfy the constraints in equation (2.8), the maximum entropy philosophy tells us to select the model that is most uniform, since entropy is highest under the uniform distribution. Therefore, the goal is to find p^* such that:

$$p^* = \arg\max_{p \in P} H(p) \tag{2.9}$$

where $H(p)$ is the entropy of the distribution p, defined as:

$$H(p) = -\sum_{x,y} p(x,y) \log p(x,y) \tag{2.10}$$

The solution to the estimation problem of finding distribution p, that satisfies the expected-value constraints, has been shown to take a specific parametric form (Berger et al., 1996) (a derivation of this parametric form is given in Appendix A):

$$p(x,y) = \frac{1}{Z} \exp(\sum_{i=1}^{m} \lambda_i f_i(x,y)) \tag{2.11}$$

with

$$Z = \sum_{(x',y') \in \Omega} \exp(\sum_{i=1}^{m} \lambda_i f_i(x',y')) \tag{2.12}$$

In more detail, f_i is the feature function (or feature for short), λ_i is the corresponding feature weight, and Z is the normalization constant that ensures that $p(x,y)$ is a proper probability distribution.

Since the sum in equation (2.12) ranges over all possible sentence-parse pairs (x, y) admitted by the grammar (all pairs in the language Ω), which is often a very large or even infinite set, the calculation of the denominator renders the estimation process computationally expensive (Johnson et al., 1999; van Noord & Malouf, 2005). To tackle this problem, a solution is to redefine the estimation procedure and consider the *conditional* rather than the joint probability (Berger et al., 1996; Johnson et al., 1999), which leads to the *conditional maximum entropy model*:

$$p(y|x) = \frac{1}{Z(x)} \exp(\sum_{i=1}^{m} \lambda_i f_i(x, y)) \tag{2.13}$$

where $Z(s)$ now sums over $y' \in \Omega(x)$, the set of parse trees associated with sentence x:

$$Z(x) = \sum_{y' \in \Omega(x)} \exp(\sum_{i=1}^{m} \lambda_i f_i(x, y')) \tag{2.14}$$

That is, the probability of a parse tree y is estimated by summing only over the parses of a specific sentence x. We can see $\Omega(x)$ as a *partition function* that divides the members of Ω into subsets $\Omega(x)$, i.e. the set of parse trees with yield x.

Let us introduce some more terminology. We will call the pair (x, y) a *training instance* or *event*. The probability $\tilde{p}(x, y)$ denotes the empirical probability of the event in the training corpus, i.e. how often (x, y) appears in the training corpus. The set of parse trees of sentence x, $\Omega(x)$, will also be called the *context* of x. By marginalizing over $\tilde{p}(x, y)$, i.e. $\sum_y \tilde{p}(x, y)$, we can derive probabilities of event contexts, denoted $\tilde{p}(x)$.

Maximum entropy models belong to the *exponential family* of models, as is visible in their parametric form given in equation (2.13). They are also called *log-linear* models, for reasons which becomes apparent if we take the logarithm of the probability distribution. It should be noted that the terms log-linear and exponential refer to the actual (parametric) form of such models, while maximum entropy is a specific way of estimating the parameters of the respective model.

The training process for a conditional maximum entropy model will estimate the conditional probability directly. This is known as *discriminative training*. A conditional MaxEnt model is therefore also known as a *discriminative model*. As before, the constraints imposed by the training data are stated as in equation (2.7), but the expectation of feature f_i with respect to a conditional model $p(y|x)$ becomes:

$$E_p[f_i] = \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y) \tag{2.15}$$

That is, the marginal empirical distribution $\tilde{p}(x)$ derived from the training data is used as approximation for $p(x)$ (Ratnaparkhi, 1998), since the conditional model does not expend modeling effort on the observations x themselves.

As noted by Osborne (2000), enumerating the parses of $\Omega(x)$ might still be computationally expensive, because in the worst case the number of parses is exponential with respect to sentence length. Therefore, Osborne (2000) proposes a solution based on *informative samples*. He shows that it suffices to train a maximum entropy model on an informative subset of available parses per sentence to estimate model parameters accurately. He compared several ways of 'picking' samples and concluded that in practice a random sample of $\Omega(s)$ works best.

Once a model is trained, it can be applied to parse selection: choose the parse with the highest probability $p(y|x)$. However, since we are only interested in the relative rankings of the parses (given a specific sentence), it actually suffices to compute the non-normalized scores. That is, we select parse y whose score (sum of features times weights) is maximal:

$$\hat{y} \quad = \quad \arg\max_{y \in \Omega(x)} score(x,y) = \arg\max_{y \in \Omega(x)} \sum_i \lambda_i f_i(x,y) \tag{2.16}$$

Parameter Estimation and Regularization

Given the parametric form in equation (2.13), fitting a MaxEnt model $p(y|x)$ to a given training set means estimating the parameters λ which maximize the conditional log-likelihood (Johnson et al., 1999):[4]

$$\begin{aligned} \hat{\lambda} &= \arg\max_{\lambda} L(\lambda) && (2.17)\\ &= \arg\max_{\lambda} \log \prod_{x,y} p(y|x)^{\tilde{p}(x,y)} && (2.18)\\ &= \arg\max_{\lambda} \sum_{x,y} \tilde{p}(x,y) \log p(y|x) && (2.19) \end{aligned}$$

[4]The following section is based on more elaborated descriptions of Johnson et al. (1999) given in Malouf and van Noord (2004), van Noord and Malouf (2005) and Malouf (2010).

This is equivalent to *minimizing* the Kullback-Leibler (KL) divergence between the model p and the empirical distribution $\tilde{p}$:

$$\begin{aligned}
D(\tilde{p}||p) &= \sum_{x,y} \tilde{p}(x,y) \log \frac{\tilde{p}(x,y)}{p(x,y)} && (2.20)\\
&= \sum_{x,y} \tilde{p}(x,y) \log \frac{\tilde{p}(x)\tilde{p}(y|x)}{\tilde{p}(x)p(y|x)} && (2.21)\\
&= \sum_{x,y} \tilde{p}(x,y) \log \tilde{p}(y|x) - \overbrace{\sum_{x,y} \tilde{p}(x,y) \log p(y|x)}^{L(\lambda)} && (2.22)
\end{aligned}$$

To find the parameters that maximize the log-likelihood $L(\lambda)$, we need to find the gradient of the log-likelihood function (i.e. the vector of its first derivatives) with respect to the parameters λ. The gradient G of $L(\lambda)$ has been shown to take the following form:

$$\begin{aligned}
G(\lambda) = \frac{\partial L(\lambda)}{\partial \lambda_i} &= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) - \sum_{x,y} \tilde{p}(x) q(y|x) f_i(x,y) && (2.23)\\
&= E_{\tilde{p}}[f_i] - E_p[f_i] && (2.24)
\end{aligned}$$

In general, there is no simple analytical solution to the problem of finding the parameters $\hat{\lambda}$. Therefore, an iterative optimization method is needed to compute model parameters. An overview of different estimation techniques for maximum entropy models is given in Malouf (2002). He shows that the *limited-memory variable metric* (LMVM) outperforms other estimation choices. This is also the default estimation technique of the *toolkit for advanced discriminative modeling* (TADM),[5] the software used to train the disambiguation component of the Alpino parser. Note that TADM was used in the experiments reported in Chapter 4. In late autumn 2010, the default estimation software of Alpino changed to `tinyest`.[6] All experiments from Chapter 5 onwards use `tinyest` for parameter estimation.

In practice, maximum likelihood estimation suffers from the problem of *overfitting*. In order to prevent the weight vector from taking arbitrarily large

[5] Available at: `http://tadm.sourceforge.net/`

[6] `tinyest` is a maximum entropy parameter estimator implemented by Daniël de Kok that additionally supports various feature selection methods. It is available at: `https://github.com/danieldk/tinyest/`

values, a *regularization* term is often added to the conditional log-likelihood function. A commonly-used method is L2 regularization, where a quadratic penalty is added for each feature weight. As shown in S. Chen and Rosenfeld (1999), this corresponds to imposing a Gaussian prior on the weights with mean zero and variance σ:[7]

$$\hat{\lambda} = \arg\max_{\lambda} L(\lambda) + \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\sum_{j=1}^{m} \lambda_j^2}{2\sigma^2}) \tag{2.25}$$

$$= \arg\max_{\lambda} L(\lambda) - \frac{\sum_{j=1}^{m} \lambda_j^2}{2\sigma^2} \tag{2.26}$$

The term $1/\sqrt{2\pi\sigma^2}$ in equation (2.25) falls away as it is constant with regard to the maximization. The gradient in (2.24) now becomes (Malouf, 2010):

$$G(\lambda)' = G(\lambda) - \frac{\sum_{j=1}^{m} \lambda_i}{\sigma^2} \tag{2.27}$$

Thus, feature expectations are constrained to equal their empirical expectation discounted by λ_i/σ^2 (S. Chen & Rosenfeld, 2000):

$$E_p[f_i] = E_{\tilde{p}}[f_i] - \frac{\lambda_i}{\sigma^2} \tag{2.28}$$

As pointed out by S. Chen and Rosenfeld (2000), this regularization technique can be seen as selecting *maximum a posteriori* (MAP) rather than maximum likelihood parameter values. In effect, the maximum likelihood estimates assume a uniform prior over the parameters (Malouf, 2010).

Another form of smoothing is to ignore certain features. This is equivalent to simply excluding some of the constraints in the model, and is also known as *constraint exclusion* (S. Chen & Rosenfeld, 2000). If we remove constraints, we will get a model with higher entropy than the original model, i.e. "a model that is smoother or more uniform" (S. Chen & Rosenfeld, 2000). This is the case because excluding features implies that we are imposing fewer constraints on the model: in the extreme case, if we have a model without any features (or, similarly, a model in which all feature weights are zero) then this model will assign equal probability to every possible outcome.[8] Thus, one

[7]We show here the regularization term with a single σ parameter, which is often used in practice. As shown in S. Chen and Rosenfeld (1999), one can also use a separate σ_j for every feature.

[8]This can be seen by inspecting the parametric formula: if either every feature weight λ_i is zero, or every feature function $f_i(x,y)$ returns 0, we will get the uniform model, since $\exp(0) = 1$.

way of smoothing is to leave out features that occur fewer than a certain number of times (specified by a threshold), i.e. to apply a *frequency cutoff* or *count cutoff*. In our task, a feature is considered relevant if it changes value at least once within $\Omega(x)$ (that is, for $y_1, y_2 \in \Omega(x)$, $f(x, y_1) \neq f(x, y_2)$). If a feature is relevant for at least t contexts (sentences), it is taken into account and is ignored otherwise. For example, a threshold of $t = 2$ means that features that are relevant for at least two sentences are included.

Minimum Divergence Models

A generalization of the maximum entropy model is the *minimum divergence* model, which explicitly includes a so-called *reference distribution* q_0 and is defined as (Berger & Printz, 1998):

$$p(y|x) = \frac{1}{Z(x)} q_0(y|x) \exp(\sum_{i=1}^{m} \lambda_i f_i(x, y)) \tag{2.29}$$

where $Z(x)$ is defined correspondingly:

$$Z(x) = \sum_{y' \in \Omega(x)} q_0(y|x) \exp(\sum_{i=1}^{m} \lambda_i f_i(x, y')) \tag{2.30}$$

The reference distribution q_0 can be used to incorporate prior knowledge into a model. As also noted in Velldal and Oepen (2005), in maximum entropy models the default model q_0 is often only implicit and not stated in the model equation, since it is assumed to be uniform, i.e. the constant function $q_0(y|s) = 1/|\Omega(s)|$. Thus, another interpretation of maximum entropy modeling is to seek a model with minimum divergence from the uniform distribution. If q_0 is uniform, then the divergence is equal to the inverse of the entropy plus a constant $D(p||q_0) = -H(x) + c$. Therefore, minimizing the KL divergence is equivalent to maximizing the entropy H (Jelinek, 1997).

If q_0 is not uniform, then p is called a *minimum divergence model* (Berger & Printz, 1998; Velldal & Oepen, 2005). In the statistical parsing literature, the default model q_0 is also referred to as base model (Berger & Printz, 1998), default or reference distribution (Hara et al., 2005; Johnson et al., 1999; Velldal & Oepen, 2005). More generally, models of this form are called *maximum entropy minimum divergence* (MEMD) models. We will return to reference distributions in Chapter 4.

2.4 Parsing Systems and Treebanks

This section introduces the three parsing systems used in this work and gives an overview of the main training corpora. While the Alpino parser, introduced next, is a grammar-driven parsing system tailored to Dutch, the two other parsing systems, MST and Malt, are two language-independent data-driven dependency parsers.

2.4.1 The Alpino Parser

Alpino (Bouma, van Noord & Malouf, 2001; van Noord & Malouf, 2005; van Noord, 2006) is a parser for Dutch that implements the conceptual two-stage parsing approach given in Figure 2.3. It has been developed over the last ten years out of a domain-specific HPSG that was developed for a spoken dialogue system (OVIS). The system is now a generic parser for Dutch.

The grammar has been augmented to produce dependency structure as output according to the LASSY format (details on the annotation scheme are given in the LASSY annotation manual).[9] The LASSY annotation scheme is largely based on the format proposed in the corpus of spoken Dutch (*Corpus Gesproken Nederlands*, CGN; Oostdijk, 2000; Hoekstra et al., 2003).[10] Figure 2.12 shows an example dependency structure in LASSY format for the following sentence:

(2.31) *Ton Sondergaard kan zijn koffers pakken*
Ton Sondergaard can his suitcases pack
'Ton Sondergaard can pack his suitcases'

Note that the LASSY format encodes dependency information between sister nodes (as further discussed below), which is a less common format to represent dependency information compared to the representation with edges between nodes (cf. the difference is illustrated in Figure 2.13). Moreover, note that the LASSY dependency structure does not necessarily reflect (surface) syntactic constituency (Bouma et al., 2001). Therefore, words are usually indexed with their position in the sentence as shown in Figure 2.12 (in the following we will omit these indices as they should be clear from the context).

[9] LASSY (Large Scale Syntactic Annotation of written Dutch).
Project website: `http://www.let.rug.nl/vannoord/Lassy/`
Annotation manual: `http://www.let.rug.nl/vannoord/Lassy/sa-man_lassy.pdf`
[10] CGN homepage: `http://lands.let.kun.nl/cgn/`

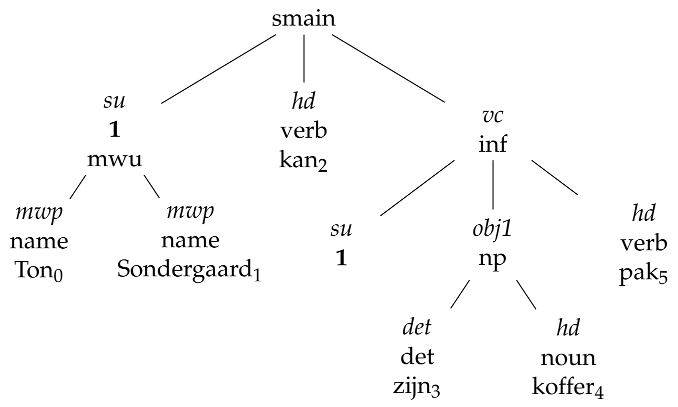


Figure 2.12: LASSY dependency structure for sentence (2.31).

The LASSY annotations contain both functional information in the form of dependency labels as well as syntactic category information. Dependency relations hold between sister nodes, where *hd* represents the head of the relation. For instance, in Figure 2.12 category information is given in the nodes, e.g. *zijn koffers* 'his suitcases' is an NP (noun phrase). Thus, the output of Alpino is somewhat a hybrid between phrase and dependency structure. Moreover, in the annotation scheme more than a single head per token is allowed. This is represented in the Alpino output by means of co-indexation. For instance, *Ton Sondergaard* (indexed with **1**) is both the subject of the auxiliary verb *kan* 'can' and the verb *pakken* 'pack'.

Alpino consists of more than 800 grammar rules in the tradition of HPSG, and a large hand-crafted lexicon. Both the lexicon and the rule component are organized in a multiple inheritance hierarchy. For words that are not in the lexicon, the system applies a large variety of unknown word heuristics, which deal with number-like expressions, compounds, proper names, etc. Lexical ambiguity is reduced by means of a PoS-tagger (Prins & van Noord, 2003). The PoS-tagger is trained on large amounts of parser output, and removes unlikely lexical categories. Some amount of lexical ambiguity remains.

A left-corner parser constitutes the parsing component, and stores the set of parses for a given sentence compactly in a packed parse forest. A best-first beam-search algorithm retrieves the best parse(s) from that forest by consulting a maximum entropy disambiguation model. The following section will introduce the maximum entropy disambiguation model of Alpino, which is the focus of the domain adaptation experiments in Chapter 4 and Chapter 5.

Maximum Entropy Disambiguation Model

The maximum entropy model employed by Alpino consists of a large set of features, corresponding to instantiations of feature templates. These features model various characteristics of parses.

To obtain the training data for the discriminative disambiguation model (i.e. possible derivations of given sentences scored by their similarity to the dependency structure in the gold-standard treebank), the training sentences are parsed with the grammar. The reason is twofold: (a) While a treebank contains gold-standard annotation (in our case dependency structures), these structures abstract away from syntactic details needed in a parse disambiguation model. In order to create a model that is sensitive to such details, the training data is obtained by parsing the training sentences. (b) By comparing the obtained parses to the gold-standard, we get access to the necessary negative training instances for the model, i.e. the incorrect parses of a sentence. Further details on the training procedure will be discussed later, we will first introduce the features used in the maximum entropy disambiguation model.

Table 2.1 gives an overview of the Alpino feature templates. In the following we discuss the major templates with the help of an example, and refer the reader for further details to van Noord (2006). Consider the sentence:

(2.32) *Het mannetje lichtte beleefd zijn hoed*
'The little man lifted cheerfully his hat'

The correct dependency structure for this sentence is shown in Figure 2.13. It shows the Alpino output (LASSY dependency structure) as well as its conversion into the more common dependency graph representation with edges between nodes. The major templates will be discussed in the following.

Description of Feature Templates The features for part-of-speech (PoS) tags are the `f1` and `f2` features, where `f2` features are more specific than the `f1` features (since they include references to the words). For example, `f1(noun)`, `f1(verb(transitive))`, `f2(man_DIM,noun)`, `f2(licht,verb(transitive))`.

Syntactic features such as the `r1` and `r2` features signal the application of a particular grammar rule in the derivation of the parse tree, with `r2` features including information from the daughter nodes (which rule was applied

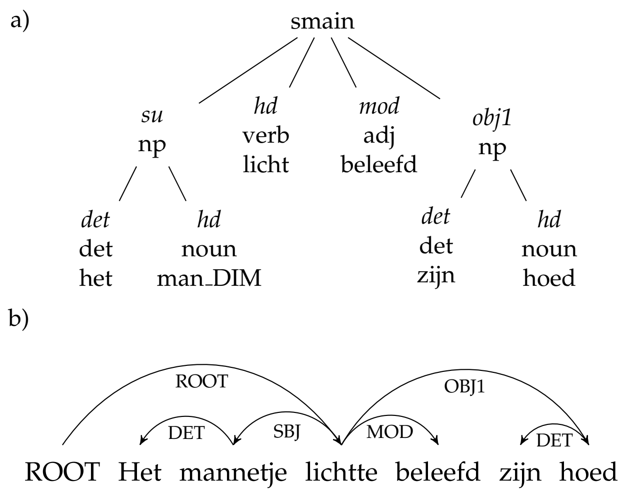


Figure 2.13: a) LASSY dependency representation (Alpino output for sentence (2.32); dependency relations hold between sister nodes, where *hd* represents the head of the relation). b) Corresponding dependency graph with edges between nodes; the arrows point from the head to the dependent (PoS tags omitted).

Description	**Feature templates**
Grammar rule applications	`r1 r2`
PoS tags	`f1 f2`
Dependency features	`dep23 dep34 dep35 depprep depprep2 sdep`
Ordering in the middle field	`mf`
Unknown word heuristics	`h1`
Selectional preferences	`z_dep35 z_depprep z_depprep2 z_appos_person`
Various other features	`appos_person s1 p1 in_year dist`

Table 2.1: Alpino feature templates.

for the n-th daughter). For example, `r1(np_det_n)` signals the application of the rule that constructs a `np` out of a determiner `det` and noun `n`. The features `r2(np_det_n,1,l)` and `r2(np_det_n,2,l)` specify that the first and second daughters of the noun phrase are lexical items `l`, i.e. `det(het,nmod)` and `noun(het,count,sg)`. In addition, there are features specifically designed to describe more complex syntactic patterns such as: orderings in the middle field (`mf` features), long-distance dependencies and fronting of subjects and other noun phrases (`dist` and `s1` features), as well as parallelism of conjuncts in coordination (`p1`).

The `dep` feature templates describe aspects of the dependency structures. For each dependency edge, different types of dependencies are extracted containing an increasing amount of information. For instance, "a noun is the subject of the verb" `dep23(noun,hd/su,verb)`, "man_DIM (diminutive of man) is the subject of the verb" `dep34(man_DIM,noun,hd/su,verb)`, and "man_DIM is the subject of the verb 'licht'" `dep35(man_DIM,noun,hd/su,verb,licht)`.

The `h1` features indicate properties of the various unknown word heuristics, for example `h1(form_of_suffix(st))`. In addition, a number of features were recently added (`z` feature templates) to include selection restrictions, estimated on the basis of large parsed corpora (van Noord, 2007). To give an idea of the number of features obtained by instantiating these templates, Table 2.2 lists the distribution of features of a model estimated from the Alpino treebank, which is described in the next section. Clearly, features that contain lexical information, such as `dep35,dep34,r2,f2` form the largest part.

Number of features per feature template

5904	`dep35`	614	`f1`	38	`z_depprep`
5433	`dep34`	411	`r1`	12	`z_depprep2`
3224	`r2`	284	`depprep2`	9	`s1`
2363	`f2`	157	`sdep`	5	`in_year`
1544	`depprep`	143	`z_dep35`	4	`z_appos_person`
1126	`dep23`	78	`appos_person`	4	`p1`
1027	`mf`	52	`h1`	2	`dist`

Table 2.2: Distribution over feature templates (with number of features instantiating the templates) for a model estimated from the Alpino (Cdb) treebank with feature cutoff $t = 2$ (described in Section 2.3.2).

Disambiguation Example Consider again sentence (2.32). Besides the correct dependency structure given in Figure 2.13, the parser might have produced an alternative, incorrect reading, whose dependency structure is similar to the one in Figure 2.13, except that the roles of two constituents is inversed: *het mannetje* is considered as direct object (obj1) and *zijn hoed* is taking the subject role.

For the disambiguation component, these two parses differ in a specific set of features, namely those shown in Table 2.3. These features *discriminate* between the two alternative readings if their feature value and weight is non-zero. As shown in Table 2.3, the parses differ in syntactic (`s1`, `mf`) and dependency relation features (`dep34`, `dep35`). The left features (corresponding to the correct reading) show that there is a topicalized subject (*man_DIM*) and an accusative noun phrase (*hoed*) that plays the direct object (`obj1`) role.

`s1(subj_np_topic)`	`s1(non_subj_np_topic)`
`mf(mcat_adv,np(acc,noun))`	`mf(mcat_adv,np(nom,noun))`
`dep34(hoed,noun,hd/obj1,verb)`	`dep34(hoed,noun,hd/su,verb)`
`dep34(man_DIM,noun,hd/su,verb)`	`dep34(man_DIM,noun,hd/obj1,verb)`
`dep35(hoed,noun,hd/obj1,verb,licht)`	`dep35(hoed,noun,hd/su,verb,licht)`
`dep35(man_DIM,noun,hd/su,verb,licht)`	`dep35(man_DIM,noun,hd/obj1,verb,licht)`

Table 2.3: Features that differ for the two alternative readings of sentence (2.32) (subject and object are inversed). Left: *het mannetje* is the topicalized subject (correct). Right: *het mannetje* is the direct object (wrong).

If we further assume that we have weights estimated for these features, then we can calculate the scores (or probabilities, i.e. normalized scores) of the two competing parses. Table 2.4 shows the relevant discriminative features, with the corresponding weights and counts (the remaining features are not contributing to disambiguation, either because they are not active or their weight is zero). In this case, the model chose the correct reading, as the score of the correct parse (with topicalized subject) is higher.

Training data: The Alpino Treebank The standard training corpus for the Alpino parser is *Cdb*,[11] the newspaper part of the Eindhoven Corpus. Cdb is a collection of text fragments from 6 Dutch newspapers (*Het Nieuwsblad van het Noorden, De Telegraaf, De Tijd, Trouw, Het Vrije Volk, Nieuwe Rotterdamse Courant*). Syntactic annotations (in XML format) are available from the Alpino treebank.[12] Therefore, we will refer to it simply as the Alpino treebank. The treebank consists of 7,136 sentences (approximately 140,000 words) with an

[11]Sometimes also called Cdbl (with final 'l'), which stands for 'Corpus dagbladen'.

[12]Available at: `http://www.let.rug.nl/~vannoord/trees/`

Correct reading:

feature	count	weight	sum
`s1(subj_np_topic)`	1	0.887404	0.887404
`mf(mcat_adv,np(acc,noun))`	1	0.158395	0.158395
		score	exp(1.045) = 2.84
		probability	0.89

Wrong reading:

feature	count	weight	sum
`s1(non_subj_np_topic)`	1	-0.970334	-0.970334
`mf(mcat_adv,np(nom,noun))`	1	-0.106108	-0.106108
`dep34(hoed,noun,hd/su,verb)`	1	0.00665973	0.00665973
		score	exp(-1.069) = 0.34
		probability	0.11

Table 2.4: Discriminative features for the two competing parses of sentence (2.32) as well as the scores and probabilities calculated.

average sentence length of 19.7 words. It is the standard treebank used to train the disambiguation component of the Alpino parsing system.

Training the Parse Disambiguation Model The training data for the disambiguation model is obtained by parsing the training sentences with the grammar. To create the training data, up to m derivations (in our experiments, either $m = 1000$ or $m = 3000$) are recorded for every sentence (context). From those, a random sample of n derivations (events) per context is taken (where $n = 100$), which forms the informative sample (Osborne, 2000, introduced in Section 2.3.2).

As the correct dependency structure might not be in this set, the parses are compared to the gold-standard dependency structure to judge their quality. Thus, every parse is assigned a score that reflects its quality in terms of similarity to the gold-standard. In our experiments, we measure the quality of a parse (roughly) in terms of the proportion of correct labeled dependency edges. The exact measure (concept accuracy) will be defined and discussed in more detail later (Section 4.3.2).

For a given sentence, the parse(s) with the highest accuracy score is (are) considered to be correct, and the other analyses are treated as incorrect. In more technical terms, the accuracy of the parse is mapped to its empirical probability $\tilde{p}(y|x)$ (as before, y is a parse and x is a sentence). This scoring of events (*event scoring*) can actually be done in various ways. One way is to treat the parses of a sentence in a binary fashion (as correct or incorrect parses), i.e. assign an event frequency of 1 to correct (highest scoring) parses and 0 to the remaining parses. These scores are then normalized such that the context probability is constant (that is, if there are multiple best parses for a context, the probability mass is divided amongst them). This is how it is implemented in the Charniak reranker parser (Charniak & Johnson, 2005).

An alternative is to give partial credit to all parses in $\Omega(x)$. That is, every parse is assigned a score that indicates its quality relative to the gold-standard. For instance, if 85% of the dependency relations are correct, the parse is considered to have a score of 0.85. This has been proposed by Osborne (2000). As he notes, the resulting distribution for $\tilde{p}(y|x)$ is not as discontinuous as the binary distribution.

Given a way to estimate $\tilde{p}(y|x)$, a related question is how contexts, the marginals $\tilde{p}(x)$, are treated. That is, whether contexts will be considered as equally likely (as before), or whether contexts that contain more probable events are considered more likely. The former approach considers the empirical frequency of all contexts equiprobable, i.e. $\tilde{p}(x)$ is estimated as $c(x)/N$, where N is the number of sentences (contexts) in the training corpus $\mathcal{D}$ and $c(x)$ is the frequency of the context in $\mathcal{D}$. This is how Osborne (2000) proposed his original event-scoring mechanism. Alternatively, van Noord and Malouf (2005) use the Osborne way of mapping scores to probabilities, but assign sentences with higher quality parses higher marginal probability (which means they treat them as more frequent).[13]

	features	contexts/sentences	events	mean events per context
Cdb	22,434	7,120	357,469	50.20

Table 2.5: Training set size for a standard model trained on the Cdb data.

After converting parse quality scores to event frequencies, a frequency-based *count cutoff* is applied (as introduced in Section 2.3.2), where $t = 2$ (unless stated otherwise). This gives the training set size shown in Table 2.5 for

[13]In the experiments of Chapter 4 we follow the approach of van Noord and Malouf (2005) that weights more probable contexts higher. As of September 2010, the standard implementation of Alpino uses normalized binary scores, as we found that gives slightly better results. This is also the approach followed for the experiments in Chapter 5.

the standard Alpino model trained on the Cdb data. Parameters of the conditional maximum entropy models are then estimated using TADM or `tinyest` (introduced in Section 2.3.2), with a Gaussian (ℓ_2) prior distribution ($\mu = 0$, $\sigma^2 = 1,000$ or $\sigma^2 = 10,000$) to reduce overfitting (S. Chen & Rosenfeld, 2000).

Once a model is trained, it can be applied to parse selection. The estimated weights determine the contribution of each feature. Features appearing in correct parses are given higher weight, while features in incorrect parses are given lower weight.

Let us consider a simple example. We are given training data that consists of two sentences (two contexts: s1 and s2), where the first sentence got 3 parses and the second sentence got 2 parses (indicated as p1, etc.). Further, assume that there are four features that describe properties of the parses (features are indexed starting from 0). Feature 0 (`f0`) takes high values for bad-quality parses, feature 2 (`f2`) is the opposite and takes high values for good parses, feature 1 (`f1`) is not active in any parse and feature 3 (`f3`) is active on all parses (events) but always got the same value. The data set is depicted as feature matrix on the left-hand side of Table 2.6, the corresponding TADM input format in shown on the right-hand side and described next.

a) Training set as data matrix:

		f0	**f1**	**f2**	**f3**
s1:	**p1**	3	-	1	1
	p2	0.1	-	2.1	1
	p3	2	-	-	1
s2:	**p1**	1	-	-	1
	p2	-	-	1.5	1

b) Corresponding TADM format:

```
3
0 3 0 3.0 2 1.0 3 1.0
1 3 0 0.1 2 2.1 3 1.0
0 2 0 2.0 3 1.0
2
0 2 0 1.0 3 1.0
1 2 2 1.5 3 1.0
```

Table 2.6: a) Rows correspond to events (s: sentence, p: parse), columns to features. Features indicated with '-' are not active in a derivation. b) TADM input format. First line: number of events that follow; An event is described as: event frequency (e.g. 1), number of features that follow (e.g. 3) and the feature-value pairs, where the feature index precedes its value. (Note that on the left all feature values are given as floats for better exposition only).

The first number of the TADM data format (which is also the input format for `tinyest`) states the number of events of the given context (sentence-parse pairs) that follow (3 in our case). Then follow the events where the first number represents the event frequency (binary or numeric scores, a lower number corresponds to worse parse quality). The second number of an event states the number of feature-value pairs that follow. For example, the first parse of the first sentence has 3 features active: feature 0 has value 3, and feature 2 and 3 have value 1. Within TADM, the raw event frequencies are normalized to obtain the proper empirical probability distributions $\tilde{p}(y|x)$ and $\tilde{p}(x)$.

Executing TADM on this data results in the following feature weights:

```
f0: -2.76947
f1: 0
f2: 3.25669
f3: -7.99557e-14
```

That is, feature 1 (`f1`) effectively got a weight of 0, as it was never active; feature 3 (`f3`) got a weight that is almost 0, as it did not distinguish any parses (had constant value for all events). Feature 0 (`f0`) got a negative weight reflecting the fact that the feature took a high feature value on bad-quality parses. On the contrary, feature 2 (`f2`) correctly obtained a high positive weight.

This ends the description of Alpino, and we will now move to two alternative parsing systems, MST and Malt. The description of these parsing systems will be relatively brief compared to the presentation of Alpino. The reason is that we mainly applied these systems as they are (out-of-the-box), unlike in the case of the Alpino parser, where we also altered parts of the system.

2.4.2 Data-driven Dependency Parsers

Two well-known dependency parsing systems are Malt[14] (Nivre et al., 2007) and MST[15] (McDonald et al., 2005). These parsing systems are purely data-driven: they do not have an explicit formal grammar. Rather, they learn their parsing model entirely from annotated data. This implies that such a parsing system can be obtained for any language for which annotated resources are available.

Both MST and Malt implement projective and non-projective parsing algorithms. Intuitively, a dependency graph is projective if the dependency graph can be drawn without crossing edges (Kübler et al., 2009; Clark, 2010). An example of a sentence that requires a non-projective dependency graph is: *A hearing is scheduled on the issue today*, taken from (Kübler et al., 2009), whose dependency graph is given in Figure 2.14. It is a non-projective dependency graph since the prepositional phrase 'on the issue' that modifies the noun

[14]We used version 1.3.1 available at: `http://maltparser.org/`

[15]We used version 0.4.3b available at: `http://mstparser.sourceforge.net/`

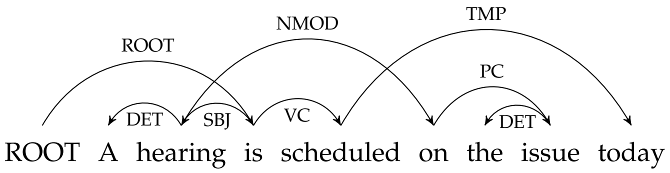


Figure 2.14: Example of a non-projective dependency graph.

'hearing' is separated from its head by the verb group in between. The resulting graph has crossing edges. In English, sentences with a non-projective analysis occur with little frequency relative to languages with fewer constraints on word order, such as for instance Czech or Dutch (Kübler et al., 2009). Therefore, we will use the non-projective parsing algorithms for our experiments on Dutch and the projective algorithm for experiments on English.

Malt and MST implement two alternative approaches to data-driven dependency parsing: Malt uses a transition-based parsing technique, while MST is a graph-based parser. In the following, we will briefly discuss each parser in turn and end with a description of the common training data format.

Malt Parser

Malt (Nivre et al., 2007) is a data-driven transition-based dependency parser. An abstract machine is employed as parsing mechanism that consists of states (configurations) and transitions between configurations. These transitions correspond to steps in the derivation of a dependency graph. The basic steps either add a dependency edge (arc) to the dependency graph or modify the state of the machine (buffer or stack, which keep the processed and unprocessed words, respectively). A classifier is used to determine the next parsing action. That is, a model is learned to score transitions from one parser state to the next. The scoring function $s(c,t)$ returns a value for transition t out of configuration c. At each stage, the model considers the parsing history and greedily takes the locally best transition out of every state, $t^* = \arg\max_t s(c,t)$, until a complete dependency graph has been created (Nivre & McDonald, 2008). Therefore, transition-based models are generally regarded as locally trained models.

The standard configuration of Malt uses a support vector machine (SVM) as classifier. Training instances for the model represent parser configurations, and the label determines the next parser action. We used the Covington non-projective parsing algorithm for the experiments with Malt on the Dutch data.

MST Parser

The MST Parser (McDonald et al., 2005) is a data-driven graph-based dependency parser. In graph-based parsing, a model is learned to score entire dependency graphs. Parsing is performed by searching for the highest scoring graph, the maximum spanning tree (MST). A separate second stage classifier is used to label the dependency edges.

In MST, a model is defined over subgraphs: the graph is decomposed (factored) into its components. Usually, arcs are considered as components, leading to arc-factored parsing models. Such a model defines a score s over arcs (Nivre & McDonald, 2008): $s(i, j, l)$, where i and j are the indices of the words between which the arc labeled l holds. The score of the parse is the sum of all arcs it contains. The goal is to find the highest scoring parse (graph G consisting of nodes V and arcs A): $G^* = \arg\max_{G=(V,A)} \sum_{(i,j,l)\in A} s(i, j, l)$. More complicated models exist. Those are parameterized over more than a single edge (and include so-called second-order features). The standard model of MST considers edges as the basic unit (first-order features). The training procedure of MST tries to maximize the global score of correct graphs. It employs an online large-margin training procedure (MIRA) as learning technique, similar in spirit to the well-known perceptron algorithm.

Experimental Setup and Training Data: CoNLL format

Both data-driven parsers (MST and Malt) are not specific to the Dutch language. Rather, they can be trained on a variety of languages given that the training corpus complies with the column-based format introduced in the 2006 CoNLL shared task (Buchholz & Marsi, 2006). An example of the CoNLL format is shown in Figure 2.15 (taken from the CoNLL 2008 data). The corresponding dependency graph is shown in Figure 2.16. A description of the data format follows.

The first column of the CoNLL data format represents the indices (IDs) of the words. The second and third column shows the words and lemmas, respectively. Column 4 and 5 represent the part-of-speech tags (which might contain coarse – column 4 – and fine PoS tags – column 5; this division is not considered in the above example). Column 6 (indicated with -) might contain

```
1       Ms.     ms.     NNP     NNP     -       2       TITLE
2       Haag    haag    NNP     NNP     -       3       SBJ
3       plays   play    VBZ     VBZ     -       0       ROOT
4       Elianti elianti NNP     NNP     -       3       OBJ
5       .       .       .       .       -       3       P
```

Figure 2.15: CoNLL data format.

additional (syntactic or morphological) information, but is left empty in the example. Column 7 indicates the ID of the head of the word (or 0 if the head is the artificial starting token). Column 8 gives the name of the dependency relation.

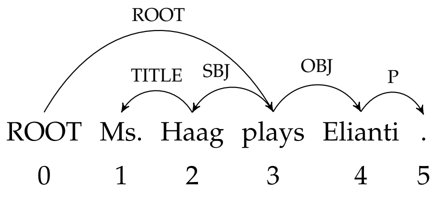


Figure 2.16: CoNLL dependency graph (with indices; PoS tags omitted).

As already mentioned, both parsing systems implement both projective and non-projective parsing algorithms. The latter will be used in our experiments on the relatively free word-order language Dutch. Moreover, both parsers expect PoS tagged data as input. Except for using the non-projective parsing mode, we train the data-driven parsers using their default settings (e.g. first-order features for MST, SVM with polynomial kernel for Malt).

2.5 Summary

This chapter provided an overview of the task of parsing and introduced the parsing systems used in this thesis. What all of these systems have in common is that there is a statistical component involved that needs training data to learn model parameters. A common problem of such *supervised machine* *learning* techniques is that they heavily depend on the data they were trained on. As a consequence, the performance of such models is impaired when they are applied to data that is different from the training data. Domain adaptation techniques try to allay this problem and are the topic discussed next.

Chapter 3

Domain Adaptation

This chapter introduces the problem of domain dependence of natural language processing systems in a general machine learning setting. The chapter provides an overview of techniques for domain adaptation that address this problem, introduces straightforward baselines and discusses prior work on domain adaptation with a special focus on natural language parsing.

3.1 Natural Language Processing and Machine Learning

The ultimate goal of natural language processing (NLP) is to build systems that are able to understand and/or produce natural language, just as we humans do. However, this is an intrinsically difficult task due to the ambiguity of natural language pertaining to all linguistic levels.

To advance NLP, an intermediate goal is to construct systems that perform well on task such as named entity recognition, part-of-speech tagging, natural language parsing, or sentiment analysis, to name but a few. Currently, to create systems that perform well on these tasks, *supervised machine learning* (ML) algorithms are employed to learn (or infer) a model capable of performing the task at hand on the basis of annotated *training data* – the general machine learning setup is illustrated in Figure 3.1. For example, in sentiment analysis the training data consists of documents (training instances) annotated with the corresponding class label (positive or negative sentiment).

Let us introduce some terminology by following the notation of Zhu and Goldberg (2009). A training instance $\mathbf{x}$ is represented by a D-dimensional feature vector $\mathbf{x} = (x_1, ..., x_D)$ where each dimension is called a feature. Note that the boldface $\mathbf{x}$ denotes a vector while x_i is a single value. The length of

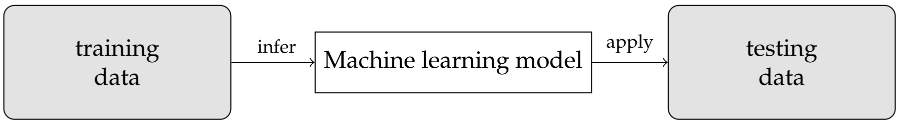


Figure 3.1: Machine learning setup.

the feature vector D is known as its dimensionality. For instance in sentiment analysis, the features might be words or sequences of words (n-grams) found in a document. The desired prediction (also known as label or class, i.e. positive or negative sentiment) of a document is denoted by y (in our example task it is an element from a finite set). The input to the learning process is the training data (or training sample), which is a collection of n training instances $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ (Zhu & Goldberg, 2009). Depending on whether label information is provided or not, machine learning algorithms fall broadly into three categories.

In *supervised machine learning* the training data contains pairs of training instances, each consisting of an instance $\mathbf{x}$ and its corresponding class label y: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. On the basis of the type of label to be learned (discrete versus continuous label), supervised machine learning can further be broadly subdivided into classification and regression, respectively. Most natural language processing tools exploit supervised machine learning.

In contrast, in *unsupervised machine learning* one is given only *unlabeled data*, i.e. training instances with no associated label: $\{\mathbf{x}_i\}_{i=1}^n$. The goal of unsupervised machine learning is usually to detect some underlying structure or pattern in the data, for instance by *clustering* the data.

A class of algorithms that is somewhere in between supervised and unsupervised learning is *semi-supervised machine learning*. In that case, the goal is to learn from both labeled and unlabeled data. In more detail, the training data consists of n labeled data instances, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and m unlabeled data instances, $\{\mathbf{x}_i\}_{i=1}^m$. It is assumed that there is considerably more unlabeled data than labeled data, i.e. $m \gg n$.

A general assumption of machine learning approaches is that the training data contains instances sampled independently from an underlying distribution: $\{\mathbf{x}_i\}_{i=1}^n \sim^{i.i.d.} p(\mathbf{x})$ (Zhu & Goldberg, 2009), where *i.i.d.* stands for identically and independently distributed. More importantly, it is assumed that the *testing data* (the data the model is tested on to check its performance or generalizability to unseen data) follows the same underlying distribution. However, this assumption is obviously violated when a system is applied to data that is different from the training data, as discussed next.

3.2 The Problem of Domain Dependence

A fundamental problem in machine learning is that supervised learning systems heavily depend on the data they were trained on. The parameters of a model are estimated to best reflect the characteristics of the training data, at a cost of portability. Indeed, the performance of a supervised learning systems drops in an appalling way when the data distribution in the training domain considerably differs from that in the test domain.

For instance, the performance of a statistical parser trained on the Penn Treebank Wall Street Journal (WSJ; newspaper text) significantly drops when evaluated on texts from other domains, as shown in Table 3.1 (note that we will delay the discussion of the notion of "domain" until Section 3.4.1 – however, intuitively, texts may differ along many dimensions related to, e.g., topic, genre and style). The Charniak parser operates on an accuracy level of around 90% when tested on data from the same domain as the training data (WSJ). However, the accuracy drops by about 6% (in absolute bracketing f-score) when the model is applied to the more varied Brown corpus (that contains fiction/non-fiction literature). It degrades by as much as 13% when it is applied to an even more distant type of text, namely abstracts from the biomedical domain (Genia) or telephone conversations from the Switchboard (SWBD) corpus.

	Test			
Train	WSJ	Brown	Genia	SWBD
WSJ	89.7	84.1	76.2	76.7

Type of Data	
WSJ:	newspaper text
Brown:	literature
Genia:	biomedical abstracts
SWBD:	phone conversations

Table 3.1: F-scores of the Charniak PCFG parser trained on the Penn Treebank WSJ and evaluated on different domains (as reported in McClosky, 2010; p.44)

The problem of domain dependence is inherent in the assumption of independent and identically distributed (i.i.d.) samples for machine learning, and thus arises for almost all NLP tasks. However, the problem has started to gain attention only in recent years. In general, there are two main solution approaches to address this problem:

1. Manually annotate data for the new domain.

2. Try to adapt a model from a specific *source domain* to a new *target domain* which is the goal of *domain adaptation*.

However, manually annotating data for the new domain leads to an unsatisfactory solution, because it is expensive and it is not a very elegant solution. In contrast, *domain adaptation* (DA) tries to adapt or port a model trained on a given *source* domain to a new *target* domain by exploiting either very limited quantities of labeled data or/and large amounts of unlabeled data from the new target domain. An example setup of domain adaptation for parsing, also known as *parser adaptation* (McClosky et al., 2006), is shown in Figure 3.2. Next, we will introduce approaches to domain adaptation.

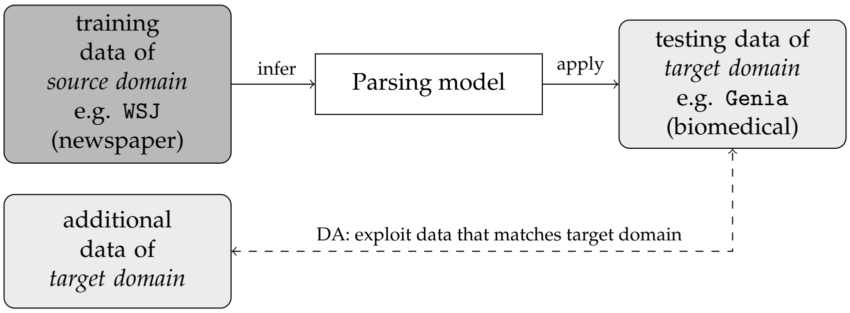

training data of *source domain* e.g. WSJ (newspaper)
infer
Parsing model
apply
testing data of *target domain* e.g. Genia (biomedical)
additional data of *target domain*
DA: exploit data that matches target domain

Figure 3.2: Domain adaptation (DA) setup. The boxes represent data sets – they are shown in different gray-scales to underline the fact that training and testing data are from different domains.

3.3 Approaches to Domain Adaptation

Similar to the general trichotomy of machine learning algorithms, there are three main approaches to domain adaptation that have been identified in the literature:

1. *Supervised domain adaptation* (e.g. Hara et al., 2005; Daumé III, 2007)

2. *Unsupervised domain adaptation* (e.g. Blitzer, McDonald & Pereira, 2006; McClosky et al., 2006)

3. *Semi-supervised domain adaptation* (e.g. Daumé III, Kumar & Saha, 2010; Chang, Connor & Roth, 2010)

The approaches differ by the kind of data that is available for the new target domain. Note that the domain adaptation literature up to 2010 generally mentioned only two main approaches: supervised and semi-supervised domain adaptation (Daumé III, 2007). That is, it was assumed that there is either only labeled or only unlabeled data available of the new domain, respectively. However, recent studies have also tried to examine scenarios in which both labeled and unlabeled data are given for the target domain (e.g. Daumé III et al., 2010; Chang et al., 2010). Therefore, there has been a shift in naming. What was previously known as semi-supervised is nowadays often called unsupervised DA, although this new 'convention' still needs to be established. In this dissertation, we follow the emerging convention.

In the following, we will discuss the three major approaches in turn and present straightforward baselines.

3.3.1 Supervised Domain Adaptation

In *supervised domain adaptation*, illustrated in Figure 3.3, we are given a rather large amount of labeled source data $D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and only a limited (comparatively small) amount of labeled data from the target domain: $D_t : \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$. That is, there is considerably more source (out-of-domain) data than target data (also known as in-domain data), i.e. $n_s \gg n_t$. The aim is to leverage the limited in-domain data together with the out-of-domain data in order to build a model that performs well on the new target domain.

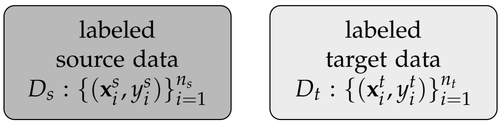

labeled
source data
$D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
labeled
target data
$D_t : \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_t}$

Figure 3.3: Supervised domain adaptation scenario. Both D_s and D_t are labeled, however $n_s \gg n_t$.

3.3.2 Unsupervised Domain Adaptation

In contrast, in *unsupervised domain adaptation*, depicted in Figure 3.4, instead of having labeled in-domain training data, we only have *unlabeled* data from the target domain. However, there is usually plenty of unlabeled target data. The goal is to exploit the original, labeled out-of-domain (source) data together with the unlabeled target data to build a model that performs well on the new target domain. This is a much more realistic situation, as unlabeled data is comparatively easy to obtain. At the same time, unsupervised domain adaptation is in general considerably more difficult.

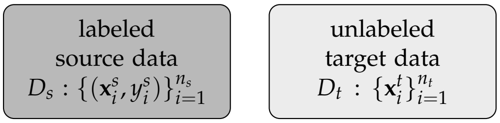

labeled
source data
$D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
unlabeled
target data
$D_t : \{\mathbf{x}_i^t\}_{i=1}^{n_t}$

Figure 3.4: Unsupervised domain adaptation scenario. Rather than having labeled data for the target domain, in this setting only unlabeled data of D_t is available. However, there might be lots of unlabeled data, i.e. $n_t \gg n_s$.

3.3.3 Semi-supervised Domain Adaptation

Only very recently studies begun to exploit both labeled and unlabeled data from the new target domain, as illustrated in Figure 3.5. Thus, the goal is to exploit the labeled source data as well as a usually rather limited amount of labeled target data together with lots of unlabeled data.

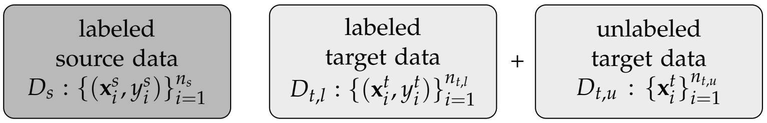

labeled
source data
$D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
labeled
target data
$D_{t,l} : \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}^{n_{t,l}}$
+
unlabeled
target data
$D_{t,u} : \{\mathbf{x}_i^t\}_{i=1}^{n_{t,u}}$

Figure 3.5: Semi-supervised domain adaptation scenario. There is both labeled and unlabeled data available for the target domain. However, there is only a small amount of labeled target data, i.e. $n_{t,u} \gg n_{t,l}$.

.

In this thesis, we will examine supervised (Chapter 4) and unsupervised domain adaptation (Chapter 5) to adapt the statistical disambiguation component of the Dutch grammar-driven parser Alpino to new domains. Adopt-ing semi-supervised DA techniques (in the new sense) is a direction for future work. Before reviewing existing literature on domain adaptation and discussing actual approaches, we first introduce straightforward baselines.

3.3.4 Straightforward Baselines

For domain adaptation, there exist several straightforward baselines:

1. Out-of-domain (source only)
2. In-domain (target only)
3. Data combination (union of source and target data; possibly weighted)

The most straightforward baseline is the *out-of-domain* – or source only – baseline: it means to train a model on the source data only, thereby ignoring any possible target data. This is the default model before applying any adaptation techniques. However, the performance of the out-of-domain model is usually rather impaired when applied on target data and thus domain adaptation aims at improving exactly this baseline model.

For the supervised domain adaptation scenario, the following baselines are also applicable: the *in-domain* – or target only – baseline and *data combination*. The in-domain baseline is a model trained on the target data only. As there is usually only a small amount of target data, the goal is to improve over this baseline by exploiting the larger amount of data from the out-of-domain (or source) domain. As we will see in the next chapter (Chapter 4) and as also discussed in Daumé III (2007) this baseline is often difficult to beat.

In contrast, *data combination* involves training a model on the *union* of the in- and out-domain data. In the literature, data combination is also referred to as 'all' (Daumé III, 2007) or 'combined' (Hara et al., 2005) baseline. Despite of its appealing simplicity, there are several disadvantages of this method. The most obvious is that it requires considerably more training time. Moreover, the large amount of out-of-domain data may 'overwhelm' the small amount of in-domain data. Therefore, a variant exists that balances the possible different amounts of in- and out-of-domain data by weighting the sources accordingly. This is also called 'weighted' in Daumé III (2007).

3.4 Literature Survey

Domain adaptation is a large area of research under constant development, with related work that can be found in different research fields (and under different names). For instance, previous work stems from both the core machine learning and the natural language processing community. Domain adaptation belongs to the more general machine learning framework of *transfer learning* (Pan & Yang, 2010), which further includes the related tasks of *multi-task learning* and *covariate shift/sample selection bias*. In general, the goal of transfer learning is to transfer knowledge across different tasks or distributions. When the goal is to transfer knowledge between different tasks, we speak about multi-task learning (e.g. Caruana, 1997). In contrast, covariate shift/sample selection bias and domain adaptation both refer to the case when distributions differ (Pan & Yang, 2010). The latter, which we are interested in here, deals with transferring knowledge in case the test and training data distributions differ. As pointed out in Dredze et al. (2007) and Jiang and Zhai (2007), theoretical work on domain adaptation attributes the performance loss to two factors: the difference in the distribution between domains (instance or feature distribution) and the difference in labeling functions. For parsing, the former corresponds to differences in the features between domains (e.g. new unseen words), while the latter may correspond to differences in annotation guidelines between domains (Dredze et al., 2007). The distribution difference is usually addressed in unsupervised adaptation approaches, while the latter is the focus of supervised approaches that require labeled target domain data. Very recent work on the new emerging field of semi-supervised domain adaptation argues that it necessary to combine these two classes of domain adaptation approaches to address both problems (Chang et al., 2010). They thus propose to exploit both labeled and unlabeled data for the new target domain, which is a setting that we have not explored yet.

In this section we review existing literature on domain adaptation with a special focus on natural language parsing. For a literature review on domain adaptation for NLP we refer the reader to Jiang (2008, chapter 6), while Pan and Yang (2010) provide a survey on transfer learning from a machine learning perspective.

3.4.1 Early Work on Parser Portability and the Notion of "Domain"

The earliest studies to examine parser portability date back to Sekine (1997), Ratnaparkhi (1999), and Gildea (2001).

Sekine (1997) argues that "it is intuitively conceivable that there are syntactic differences between 'telegraphic messages' and 'press reports' [...]". He was the first to carry out an empirical investigation that compares structure distributions (of CFG rules) in the subdomains of the Brown corpus. Related to this is the work by Roland and Jurafsky (1998), who found that verb subcategorization frames are different between the Brown, WSJ and Switchboard corpora. For example, they found that the verb 'to pass' is mainly used in the movement sense in Brown, e.g. "Blue Throat's men spotted him .. as he passed". In contrast, the legislative (or law) sense was most prominent in the WSJ, e.g. "Britain's House of Commons passed a law that will force English soccer fans to carry identity cards to enter stadiums".

While Sekine (1997) analyzed parser performance within subdomains of the Brown corpus, to our knowledge, Ratnaparkhi (1999) and Gildea (2001) were the first to examine the effect of parsing accuracy across corpora (between WSJ and Brown). The obvious motivation was:

> Most work in statistical parsing has focused on a single corpus: the Wall Street Journal portion of the Penn Treebank. While this has allowed for quantitative comparison of parsing techniques, it has left open the question of how other types of text might affect parser performance, and how portable parsing models are across corpora. (Gildea, 2001)

Both Ratnaparkhi (1999) and Gildea (2001) analyzed the performance of a phrase-structure parser trained on the WSJ when applied to the Brown corpus. They found that the accuracy decreases by around 6% (in absolute f-scores) when tested on Brown, and simple data combination derived only a small benefit over just using in-domain Brown data – Table 3.2 shows their results (Gildea, 2001).

Training Data	**Test Set**	**F-score**
WSJ	WSJ	86.34
WSJ	Brown	80.64
Brown	Brown	84.09
WSJ+Brown	Brown	84.33
WSJ+Brown	WSJ	86.59

Table 3.2: Parsing results from Gildea (2001) by training and test corpus. Size of training sets, respectively: WSJ 39,832 sentences; Brown 21,818 sentences.

Thus, one conclusion from that line of work is that a large amount of out-of-domain data does not help: "Our results show strong corpus effects for statistical parsing models: a small amount of matched training data appears to be more useful than a large amount of unmatched data" (Gildea, 2001). In addition, Gildea (2001) analyzed the effect of including lexical dependencies. He found that a large part of the parameter space of his parser can be removed without much effect on accuracy. A model with and without lexical dependency features resulted in almost identical performance on the target Brown data. On the source domain data the performance dropped by less than 0.5% f-score.

While Ratnaparkhi (1999) and Gildea (2001) both focus on the WSJ and Brown corpora, Lease and Charniak (2005) were the first to analyze parser portability to another domain: biomedical data (Genia). They show that the unknown word rate increases as they move to increasingly technical domains, and that this clearly affects parsing accuracy. Rather than trying to adapt the parsing system itself, they focus on the adaptation of pre-processing steps such as the PoS tagger. This is similar to Rimell and Clark (2008), who studied the effect of adapting the PoS tagger and supertagger of their CCG parser to contrasting domains (biomedical data and questions).

In general, the observed performance loss on various domains is an indication that statistical parsers are sensitive to data shifts. One possible approach to address this problem is to apply regularization techniques, i.e. to integrate prior knowledge into a model. This will be discussed in the next section. However, before moving to actual adaptation approaches, let us first examine the notion of "domain" as perceived in the literature and look at text excerpts to get an intuitive feeling on how texts differ.

The Notion of "Domain" and Text Excerpts from Various Sources The categories found in the Brown corpus are considered domains in Sekine (1997) and Ratnaparkhi (1999). They include "general fiction", "romance and love story", "press: reportage". In contrast, Gildea (2001) does not explicitly mention the term domain. However, he speaks about "text types" and "different genres of text". Similarly, Lease, Charniak, Johnson and McClosky (2006) mention the problem of portability in the context of differences across genres: "As might be expected, a parser trained on one genre of text (e.g. WSJ) tends to perform less accurately when evaluated on a different genre." (Lease et al., 2006). Blitzer et al. (2006) attribute domain differences mostly to differences in vocabulary, while Biber (1988) explores differences between corpora from a sociolinguistics perspective. For parsing, McClosky (2010) states that the problem of parser portability "is usually attributed to a difference in *domain*. By domain, we mean the style, genre, and medium of a document." Thus, there is no common ground on what constitutes a domain – moreover, terms such as genre, register, text type, domain, style are often used differently in different communities (D. Y. Lee, 2001).

Intuitively, texts vary along several dimensions or parameters, as they are called in Karlgren and Cutting (1994). To give an impression of the variability of language across texts, Figure 3.6 on page 51 and Figure 3.7 on page 52 show text excerpts from various sources (for Dutch and English, respectively). Note that the data is tokenized (split into separate sentences and punctuation is separated from word tokens) which is a normal pre-processing step for parsing.

Newspaper text (e.g. WSJ, Cdb) contains relatively long sentences from the written register, that clearly follows standard practices of written language. In contrast, the spoken data contains many idiosyncrasies of spoken language, for example, hesitation, repetition, false starts, contracted forms (e.g. *'k* instead of *ik* 'I'), extra-linguistic signals (e.g. pauses). Similarly, social media posts (e.g. tweets from Twitter) contain many contractions (e.g. *kga* standing for *ik ga* 'I go'), but additionally also include smiley's, spelling errors (*spekgald* rather than *spekglad* 'slippery'), URLs and special tokens expressing mood or user names. This type of data might be considered even more distant from newspaper text than the spoken data.

One simple (to compute) and obvious dimension of variability is sentence length. Figure 3.8 plots the sentence length distribution of data from various sources. For instance, the figure on the left shows that the sentence length distribution of the spoken data (CGN) is rather different from that of newspaper text (Cdb). The spoken data contains a large fraction of very short sentences. Similarly, for the English data, the plot on the right shows that the average sentence length in Brown is shorter than that in the WSJ. Thus, the sentence length distribution may reveal differences between text collections. On the other hand, the sentence length distributions of the WSJ and Genia corpus are comparatively similar – by looking only at the location and shape of the distribution (and ignoring actual frequency) we may not be able to distinguish between them. Thus, there is more than one dimension that accounts for differences between texts. Intuitively, by looking at the text excerpts in Figure 3.7, the biomedical text from Genia contains a wealth of highly specialized terms, and therefore vocabulary differences might account for a considerable part of the domain difference between WSJ and Genia. This is supported by the results of Rimell and Clark (2008). They found that, for parser performance, retraining the PoS tagger accounted for a greater proportion of the improvement on the biomedical data, while retraining the supertagger (that includes subcategorization information) was more beneficial for the questions domain. Thus, intuitively, they argue that the main difference between newspaper and biomedical text is in vocabulary, while the main difference between newspaper text and questions is syntactic. Indeed, questions and biomedical data are presumably rather different types of data, however, it will be less clear what is more beneficial for, say, biomedical questions. In general, both lexical

Newspaper text (Cdb):

```
De jaarlijkse omzet van Xiat bedraagt ongeveer flo 4 miljoen .
Xiat heeft zijn hoofdkantoor in Villanova d'Ardenghi bij Pavia , alsmede een
verkoopkantoor in Milaan .
De militairen Van Wingerden en Steenbeek zijn op 12 mei 1940 in Breda ten
onrechte doodgeschoten .
Beide militairen werden in de meidagen van 1940 op de binnenplaats van het
Bredase politiebureau zonder vorm van proces gefusilleerd door een sergeant ,
die daartoe van de sergeant-majoor H. de Leeuw opdracht had gekregen .
```

Spoken data (CGN):

```
heb je niet gekookt ?
ik had geen zin om te koken bah .
en uhm ...
ik had nog een restje vlees dat heb 'k afgeknabbeld .
waren van die spareribs .
oh lekker .
en uh ...
toen heb 'k wat vla gegeten .
toen had ik nog honger .
toen heb 'k een stukje brie gegeten .
ggg .
wat een combinatie .
```

Social media data (Twitter):

```
Wilt iemand een vespa 125cc ?
RT @bkkc_nieuws : Het bkkc heeft een nieuwe facebookpagina !
#voelmekut http://bit.ly/iOYRNE
wat doet een douche veel goeds zeg , gelijk weer stuk fitter :)
=S / dat zegik tog xd
Ok , kga nu slapen .
ik bedoelde ... " and then you meet that one person " ....
@sophievh en k mag niet meer pinge van mn moeder trouwens xD
IK WIL NOG STEEDS BROODJE EI
1234 is best een moeilijke code
Spekgald buiten !
@mennooow ik heb al 23 tweets #weinig
```

Figure 3.6: Excerpts of Dutch data from various sources.

Newspaper text (WSJ):

```
Rolls - Royce Motor Cars Inc. said it expects its U.S. sales to remain steady at
about 1,200 cars in 1990 .
The luxury auto maker last year sold 1,214 cars in the U.S.
Howard Mosher , president and chief executive officer , said he anticipates
growth for the luxury auto maker in Britain and Europe , and in Far Eastern
markets .
Bell , based in Los Angeles , makes and distributes electronic , computer and
building products .
Investors are appealing to the Securities and Exchange Commission not to limit
their access to information about stock purchases and sales by corporate
insiders .
```

Literature (Brown):

```
With the end of the trial Diane disappeared from New York .
Several years ago she married a Houston business man , Robert Graham .
She later divorced Graham , who is believed to have moved to Bolivia .
Houston police got to know Diane two years ago when the vice squad picked her up
for questioning about a call girl ring .
The next time the police saw her she was dead .
It was September 20 , 1960 , in a lavishly decorated apartment littered with
liquor bottles .
She had a party with a regular visitor , Dr. William W. McClellan .
When the police arrived , they found McClellan and the two lawyers sitting and
staring silently .
An autopsy disclosed a large amount of morphine in Diane ’s body .
‘‘ I think that maybe she wanted it this way ’’ , a vice squad cop said .
```

Biomedical abstracts (Genia):

```
Glucocorticoid resistance in the squirrel monkey is associated with
overexpression of the immunophilin FKBP51 .
The low binding affinity of squirrel monkey GR does not result from
substitutions in the receptor , because squirrel monkey GR expressed in vitro
exhibits high affinity .
Rather , squirrel monkeys express a soluble factor that , in mixing studies of
cytosol from squirrel monkey lymphocytes ( SML ) and mouse L929 cells , reduced
GR binding affinity by 11-fold .
```

Figure 3.7: Excerpts of English data from various sources.

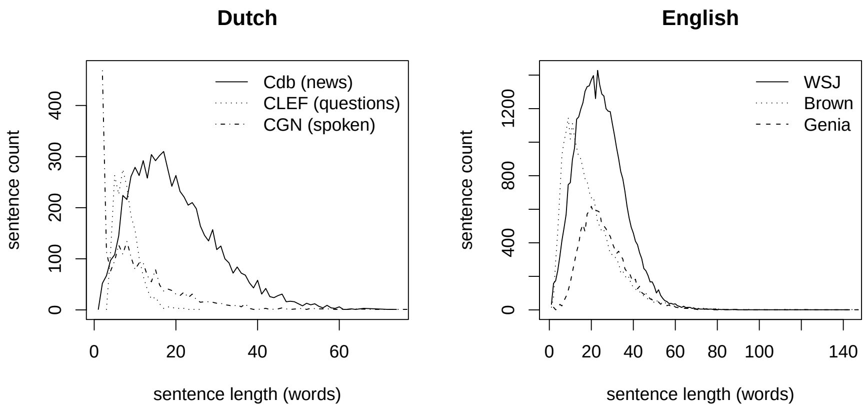


Figure 3.8: Sentence length distribution of data from various sources.

and syntactic differences will play an important role between domains. This brings us to our next point, namely, what in practice has been considered as *domain*.

In general, most previous work on domain adaptation used the term *domain* somewhat arbitrarily. We here quote from Finkel and Manning (2009):

> The word *domain* is used here somewhat loosely: it may refer to a topical domain or to distinctions that linguists might term mode (speech versus writing) or register (formal written prose versus AMS communications).

Practically, a *domain* is defined by the corpus – thus, a corpus constitutes the domain, e.g. the WSJ domain or the Genia domain. However, this is not quite a universal view – a corpus might contain one domain, but might also consist of several subdomains (e.g. Sekine, 1997; Kilgarriff & Rose, 1998; Plank & Sima'an, 2008; Webber, 2009; Lippincott, Ó Séaghdha, Sun & Korhonen, 2010). The first even argues that "when we try to parse text in a particular domain, we should prepare a grammar which suits that domain" (Sekine, 1997).

For the first part of this thesis, we will follow the general trend of considering a particular corpus as a text domain. However, we will come back to this issue in the last chapter (Chapter 7), in which we devise a measure of domain similarity for parsing.

We want to stress the fact that previous work often associated *domain* with shifts in topics only – this is not the view taken here. Instead, we consider a domain as necessarily multi-dimensional, and thus we do not want to equate it with any particular term. Rather, let us consider the term *domain* as hypernym spanning over all kinds of variability (and their interplay) between texts, e.g. topic, genre, style, medium, to name but a few. Obviously, we delimit the variability to that found within a particular language, as we will examine parser portability on Dutch and on English, but not across languages.

3.4.2 Overview of Studies on Domain Adaptation

In this final section, we will review previous studies on domain adaptation. In the first part, we will discuss studies that focused on supervised domain adaptation (where some limited amount of labeled target data is available), while in the second part we present prior work on using unlabeled data.

Prior Work on Supervised Domain Adaptation

Previous studies that focus on supervised domain adaptation include Roark and Bacchiani (2003), Hara et al. (2005), Daumé III (2007) and Jiang and Zhai (2007).

One way to bridge the gap between in-domain and out-of-domain data is to incorporate *prior knowledge* into a model. This is basically the approach taken in Roark and Bacchiani (2003), Hara et al. (2005) and Chelba and Acero (2006). That is, the original, out-of-domain, source domain model is exploited as a prior when estimating a model on the target domain for which only limited resources are available. Roark and Bacchiani (2003) examined this idea to adapt a PCFG parser, while Chelba and Acero (2006) successfully applied it to another task (automatic capitalization). In contrast, Hara et al. (2005) try to adapt the probabilistic disambiguation component of a grammar-driven parser (based on a HPSG) trained on newspaper text (WSJ) to the biomedical domain (GENIA corpus). The disambiguation component of their parser, Enju, is a maximum entropy model on a packed forest structure. Their approach consisted in integrating the original model estimated on the larger, out-of-domain data (newspaper text) as a *reference distribution* (introduced on page 26) when learning a model on the target domain. Their empirical results, also given in Table 3.3 below, show that incorporating the general model helped in outperforming the in-domain baseline. However, simple data combination came close to the result obtained by their method. Note that they only had a very small amount of in-domain data available (the Genia corpus as of today contains approximately 14,000 annotated sentences, thus it is four times the size, which will render it a harder baseline). In the next chapter (Chapter 4) we evaluate a similar approach to supervised domain adaptation by exploiting auxiliary distributions. Therefore, we will return to the reference distribution-based approach.

	F-score
Reference distribution (Hara et al., 2005)	86.87
Data combination	86.32
Genia only (in-domain baseline)	85.72
WSJ, original (out-of-domain) model	85.10

Table 3.3: Results reported in Hara et al. (2005), where the general model is exploited as reference distribution for the target domain. The size of the Genia and WSJ corpus is, respectively: 3,524 and 39,832 sentences.

Another approach to domain adaptation is to *alter the feature space*. A study that exclusively focused on the supervised scenario and changed the feature space is Daumé III (2007). In his paper he introduces an algorithm called *easy adapt*. The idea of *easy adapt* is to transform the domain adaptation learning problem into a standard supervised learning problem to which "any standard algorithm may be applied (e.g. SVM, MaxEnt)" (Daumé III, 2007). In order to achieve this, he proposes a simple pre-processing step in which the feature space is altered and the resulting data is used as input to a standard learning algorithm. In more detail, *easy adapt* basically triples the feature space: it takes each feature in the original feature space and makes three versions of it, a general version, a source-specific and a target-specific version. By transforming the feature space, the supervised learning algorithm is supposed to 'learn' which features transfer better to the new domain. Using this simple transformation, Daumé III (2007) achieved decreased error rates on several datasets and tasks (e.g. named entity classification, PoS tagging). As stated in the paper, a precondition for the pre-processing technique to work is that "one has enough labeled target domain data to do slightly better than just using only source data" (Daumé III, 2007). We will explore his approach in Chapter 4 (Section 4.2).

Note that *easy adapt* is actually a simplified version of a more complicated model proposed earlier (Daumé III & Marcu, 2006). As they show in Daumé III (2007), *easy adapt* performed similar to or better than their previously (rather involved) suggested approach (which is based on hidden variables and the expectation maximization learning algorithm), thus we will not discuss it here further. A recent study related to *easy adapt* is Finkel and Manning (2009). They extend *easy adapt* by adding explicit hyperparameters to the model. That is, each domain has its own set of domain-specific parameters, but they are linked by introducing an additional layer that acts as general parameter set. This parametrization encourages the features to have similar weights across domains, unless there is evidence in the domains to the contrary (Finkel & Manning, 2009). They showed that this extension outperformed the feature duplication approach of Daumé III (2007).

A basic assumption of these three approaches is that there are three underlying distributions (rather than just two): an in-domain (or target) distribution, an out-of-domain (or source domain) distribution and a general domain distribution. Thus, it is assumed that the in-domain data is drawn from a mixture of an in-domain and a general distribution. Similarly, the out-of-domain data is drawn from a out-of-domain and the general distribution. They motivate their choice with an example coming from part-of-speech tagging: "For example, the assignment of the tag 'determiner' (DT) to the word "the" is likely to be a *general* decision, independent of domain. However, in the Wall Street Journal, "monitor" is almost always a verb (VB), but in technical documentation it will most likely be a noun." (Daumé III & Marcu, 2006).

An alternative approach is to *change the instance distribution*, rather than altering the feature space. This has been suggested as *instance weighting* in Jiang and Zhai (2007). They propose to weigh source training instances by their similarity to the target distribution. They have shown the effectiveness of this approach on three NLP tasks (PoS tagging, named entity classification and spam filtering). In the last section of their paper, they additionally evaluated a *bootstrapping* scenario in which they added the most confidently predicted unlabeled target instances with their predicted label to the training set. This brings us directly to the next section.

Studies on Unsupervised Domain Adaptation

In contrast to the supervised scenario, studies on *unsupervised domain adaptation* exploit unlabeled data from the target domain, besides the limited amount of labeled data from the source domain only. Unsupervised domain adaptation approaches include *bootstrapping*, like self-training (McClosky et al., 2006) and co-training (Steedman et al., 2003). An alternative approach is to infer a *shared feature representation* between domains (Blitzer et al., 2006). In the following, we will discuss these approaches.

A common approach of using unlabeled data is *bootstrapping*, which is a general semi-supervised machine learning approach. A model is trained on the labeled data and is then applied to label data from a pool of unlabeled data instances. A new model is then trained on the combination of the original labeled and the automatically labeled data, and this process might be iterated. Note that in general in machine learning this is referred to as semi-supervised learning (as there is both labeled and unlabeled data), while in domain adaptation we will refer to this specific setting as unsupervised DA (in order to distinguish it from the semi-supervised domain adaptation setting in which we have both labeled and unlabeled data from the target domain).

In *self-training* a model is used to label data for itself. In contrast, in *co-training* there are two (or more) models, where one model is labeling data for the other. An assumption of co-training is that each model has a different view on the task. Steedman et al. (2003) explore co-training for bootstrapping statistical parsers. They used a lexicalized PCFG parser and a lexicalized tree adjoining grammar (LTAG) parser as their two base parsers. They performed experiments in which the parsers are trained on small seeds of labeled data (500 sentences). Then, either co-training or self-training is used to automatically label data. They showed that co-training is beneficial when the seed set is small, that co-training outperformed self-training, and that co-training is helpful in the case the seed data is from a different domain than the unlabeled data (in their case, WSJ and Brown). A scheme similar to co-training was used by Sagae and Tsujii (2007). They used two different parsing models to parse unlabeled data of the target domain. Subsequently, they added only those sentences to the source training data for which the two models produced identical analyses. This approach helped them to improve parsing accuracy, and their system actually scored highest in the 2007 CoNLL shared task on domain adaptation.

McClosky et al. (2006) and McClosky and Charniak (2008) applied self-training to the generative Charniak PCFG parser. While it was generally assumed that self-training does not help (e.g. Charniak, 1997; Steedman et al., 2003), they were the first to show that self-training can help for improved parsing accuracy when used in combination with a second-stage discriminative parse reranker. They used self-training with a large seed set and tested it on the Brown corpus (McClosky et al., 2006) and biomedical data (McClosky & Charniak, 2008). Reichart and Rappoport (2007) were the first to show that self-training (without reranking) can also be effective in case the seed set is small (in contrast to Steedman et al. (2003) they added the automatically labeled data all at once rather than adding a small amount at each iteration). Moreover, very recently, Sagae (2010) evaluated self-training with and without reranking and its impact on semantic role labeling. He showed that despite the fact that simple self-training (with no reranker) does not always improve parser performance, it can be effective when evaluated on a subsequent task that relies on the parser's output. However, it is still necessary to investigate the reason why, for instance, simple self-training (without reranking) proved to be better for adapting a semantic role labeler than self-training with reranking. Thus, due to these mixed results, bootstrapping remains a challenging and active area of research.

Another line of work that focuses on unsupervised domain adaptation is Blitzer et al. (2006). They propose *structural correspondence learning* (SCL). It is an algorithm that tries to exploit unlabeled data from both source and target domains to learn a common feature representation that is meaningful across domains. The key idea of SCL is to automatically identify correspondences among features from different domains by modeling their correlations with so-called *pivot features*. Pivots are features occurring frequently in both domains. Non-pivot features that correspond with many of the same pivot-features are assumed to correspond. SCL has proven to be successful for the two tasks examined, PoS tagging and sentiment classification (Blitzer et al., 2006; Blitzer, Dredze & Pereira, 2007). An attempt was made in the CoNLL 2007 shared task on domain adaptation to apply SCL to data-driven dependency parsing (Shimizu & Nakagawa, 2007). However, their system just ended up on place 7 out of 8 teams due to a bug in their system. Moreover, as discussed in Dredze et al. (2007), the biggest problem of the shared task was that the datasets provided were annotated with different annotation guidelines. Therefore, results are inconclusive and structural correspondence learning remains rather unexplored for parsing. In Chapter 5, we will present an application of SCL to the statistical disambiguation component of the Alpino parser and we will compare it to simple self-training.

This literature review is necessarily incomplete as the literature on domain adaptation is huge. For instance, just for the task of parsing there are other approaches that we did not mention above (e.g. Foster, Wagner, Seddah & Genabith, 2007; W. Chen, Wu & Isahara, 2008; Zhang & Wang, 2009; McClosky, Charniak & Johnson, 2010). However, we tried to give an overview of the field by focusing on approaches that we consider relevant to our work. The work described here paves the way for the next two chapters, while studies that are relevant for later chapters will be discussed then (i.e. Zhang & Wang, 2009; McClosky et al., 2010).

We want to conclude this chapter with a summary and pointers to recent work that address problems that have mostly been neglected so far, but we believe that they are relevant for the practical advance of domain adaptation.

3.5 Summary and Outlook

The goal of this chapter was to introduce the problem of domain dependence and provide an overview of approaches to tackle this problem, commonly known as domain adaptation techniques. Depending on the type of data that is available for the target domain, there are three main approaches: supervised, semi-supervised and unsupervised domain adaptation.

The chapter provided a literature overview of current approaches to domain adaptation with special attention to natural language parsing. Moreover, we discussed the notion of *domain* and noted that it was mostly arbitrarily used to refer to some kind of coherent unit. In practice, a domain is defined by the corpus given. This implies that most previous work on domain adaptation focused on adapting a system trained on one specific domain (say, the WSJ) to a particular other domain (say, the Genia corpus). In this setting, one knew that a domain change had occurred, knew what the source and what the target domain was and, additionally, had data available for that domain to exploit. This is also the scenario we will assume in Chapter 4 and Chapter 5.

However, as more and more data becomes available, we believe that it becomes increasingly important to address the related issues of *domain detection* and *domain selection*. For instance, what is the best way to use available sources (e.g. which data or model(s) should be used to parse an unknown target domain?). Similarly, how can we detect a domain shift without having access to annotated data? Quoting Van Asch and Daelemans (2010), as of now:

> There is unfortunately no unambiguous measure to assert a domain shift, except by observing the performance loss of an NLP tool when applied across different domains. This means that we typically need annotated data to reveal a domain shift.

However, can we detect a domain shift without having access to annotated data? This could be useful, for example, to trigger adaptation techniques. A first study that goes in this direction is the recent work by Dredze, Oates and Piatko (2010). They use the $\mathcal{A}$-distance (Kifer, Ben-David & Gehrke, 2004) to detect shifts in online streaming data. The measure has been used in a batch setting by Daumé III and Marcu (2006) and Blitzer et al. (2007), who train a linear classifier to discriminate between instances from different domains. In Chapter 7 we will devise a simple measure of domain similarity to select related training data for a new target domain. The work most related to this is the recent study by McClosky et al. (2010). They address the problem of finding the best mixture of parsing models trained on several source corpora to parse an unknown target text. We will discuss their work in further detail in Chapter 7.

For the practical success of parser portability, we believe that the adaptation of pre-processing tools like PoS taggers or tokenizers as well as the lexical component of grammar-driven parsers will be important, too. However, we mainly focus on parser adaptation in this dissertation. More specifically, we will examine the adaptation of the syntactic disambiguation component of a grammar-driven parser in the next two chapters, while the last chapter focuses on domain adaptation for a data-driven dependency parser.

Part II

Domain Adaptation of a Syntactic Disambiguation Model

Chapter 4

Supervised Domain Adaptation

In this chapter we focus on supervised domain adaptation and investigate the application of *auxiliary distributions* (Johnson & Riezler, 2000) for domain adaptation of the syntactic disambiguation model of the Alpino parser (introduced in Chapter 2). We compare the approach to a range of straightforward baselines and prior work, and evaluate it on two application domains: question answering and spoken data. The result is negative: the auxiliary-based approach does not work for domain adaptation; better results are achieved either without adaptation or by simple model combination.[1]

4.1 Introduction and Motivation

For parsing, most previous work on domain adaptation has focused on *data-driven* systems (Gildea, 2001; Roark & Bacchiani, 2003; McClosky et al., 2006; Shimizu & Nakagawa, 2007), that is, systems that automatically induce their grammar or model from an annotated corpus (Chapter 2).

Parse selection (also called parse disambiguation or parse reranking) constitutes an important part of many parsing systems (Johnson et al., 1999; Hara et al., 2005; van Noord & Malouf, 2005; McClosky et al., 2006). Yet, the adaptation of parse selection models to novel domains is a far less studied area. This may be due to the fact that potential gains for this task are inherently bound by the underlying grammar.

In this and the following chapter, we focus on domain adaptation of parse selection models. More specifically, we focus on *supervised domain adaptation* in this chapter, and explore *unsupervised domain adaptation* approaches (that

[1]A part of this chapter is published in Plank and van Noord (2008).

use only unlabeled target domain data) in the next chapter. As introduced in Section 3.3, supervised domain adaptation is the scenario in which we have access to some – but a limited amount of – labeled data of the new target domain. Thus, given a parsing system and two data sets, the original (out-of-domain or simply OUT) source data and the (new or IN-domain) target data, the goal is to adapt the parsing system (that is by default trained on the out-of-domain data) to the new target domain.

For the empirical evaluations of this and the following chapter, we use Alpino, a natural language parser for Dutch. Alpino employs a discriminative approach to parse selection that bases its decision on a conditional maximum entropy (MaxEnt) model. An introduction to the MaxEnt framework and the Alpino parser can be found in Section 2.3 and Section 2.4.1, respectively.

In particular, we investigate the use of *auxiliary distributions* (Johnson & Riezler, 2000) for domain adaptation, originally suggested for the incorporation of lexical selectional preferences into a parsing system. We exploit an original and larger out-of-domain model as auxiliary distribution (Section 4.2), and compare it to a range of straightforward baselines (introduced in Section 3.3.4) and prior work (Section 4.5). The approach is examined on two application domains: question answering and spoken data.

Our empirical results show that the auxiliary distribution does not help. The incorporation of the out-of-domain (source) model as auxiliary feature(s) in the target domain model does not contribute to improved parsing accuracy on the new domain – even in the case only very little target training data is available; instead, better results are achieved either without adaptation or by simple model combination.

4.2 Auxiliary Distributions for Domain Adaptation

First of all, auxiliary distributions will be introduced and defined. Subsequently, we will show how to apply auxiliary distributions to adapt a discriminative parse selection model to novel domains.

4.2.1 What are Auxiliary Distributions

Auxiliary distributions are a mechanism to incorporate information from additional, "auxiliary" sources into a maximum entropy model. In more detail, auxiliary distributions are integrated into a model by considering the logarithm of the probability given by an auxiliary distribution as an additional, real-valued feature.

More formally, given a conditional maximum entropy model $p(\omega|s)$ (i.e. equation (2.13), restated below) that consists of m feature functions f_i, where ω and s represent a parse tree and sentence, respectively:

$$p(\omega|s) = \frac{1}{Z(s)} \exp(\sum_{i=1}^{m} \lambda_i f_i(\omega, s)) \tag{4.1}$$

And, given k auxiliary distributions $q_1, ..., q_k$ that model aspects of $p(\omega|s)$. Then, k new *auxiliary features* $f_{m+1}, ..., f_{m+k}$ are added to the model, such that:

$$f_{m+i}(\omega, s) = \log q_i(\omega|s) \tag{4.2}$$

The auxiliary distributions $q_i(\omega|s)$ do not need to be proper probability distributions, however they must strictly be positive $\forall \omega \in \Omega(s)$ (Johnson & Riezler, 2000). $\Omega(s)$ is the set of parses of sentence s.

Auxiliary distributions resemble a reference distribution (cf. Section 2.3.2), but instead of considering a single reference distribution they have the advantage that several auxiliary distributions can be integrated and weighted against each other. Johnson and Riezler (2000) establish the following equivalence between the two and show that a reference distribution is a geometric mixture of k auxiliary distributions:

$$q(\omega|s) = \prod_{i=1}^{k} q_i(\omega|s)^{\lambda_{m+i}} \tag{4.3}$$

where $q(\omega|s)$ is the reference distribution, i.e. q_0 in equation (2.29), and $q_i(\omega|s)$ is an auxiliary distribution. The contribution of each auxiliary distribution $q_i(\omega|s)$ is regulated through the estimated feature weight λ_{m+i}, which allows the feature to get a discounted weight or even be ignored.

In general, a conditional maximum entropy model that includes k auxiliary features takes the following form (Johnson & Riezler, 2000):

$$p(\omega|s) = \frac{\prod_{i=1}^{k} q_i(\omega|s)^{\lambda_{m+i}}}{Z(s)} \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega, s)) \tag{4.4}$$

with Z(s):

$$Z(s) = \sum_{\omega' \in \Omega(s)} \prod_{i=1}^{k} q_i(\omega|s) \exp(\sum_{i=1}^{m} \lambda_i f_i(\omega', s)) \tag{4.5}$$

Due to the equivalence relation in equation (4.3) we can restate equation (4.4) to show explicitly that auxiliary distributions are regarded as additional features. Note that the step from equation (4.4) to (4.6) holds by restating equation (4.2) as $q_i(\omega|s) = \exp(f_{m+i}(\omega,s))$.

$$p(\omega|s) = \frac{\prod_{i=1}^{k} [\exp(f_{m+i}(\omega,s))]^{\lambda_{m+i}}}{Z(s)} \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega,s)) \tag{4.6}$$

$$= \frac{1}{Z(s)} \exp(\sum_{i=1}^{k} f_{m+i}(\omega,s)\lambda_{m+i}) \exp(\sum_{j=1}^{m} \lambda_j f_j(\omega,s)) \tag{4.7}$$

$$= \frac{1}{Z(s)} \exp(\sum_{j=1}^{m+k} \lambda_j f_j(\omega,s)) \quad \text{with } f_j(\omega,s) = \log q_j(\omega|s) \text{ for } m < j \leq (m+k) \tag{4.8}$$

Johnson and Riezler (2000) introduced auxiliary distributions to incorporate lexical selectional preferences estimated from a larger corpus into a stochastic attribute-value grammar (SAVG). They used a simpler syntactic structure (shallow parses) to parse a large corpus which forms the basis to estimate their auxiliary distributions. They integrated these additional features into the SAVG model to overcome data sparseness.

4.2.2 Exploiting Auxiliary Distributions for Domain Adaptation

While Johnson and Riezler (2000) use auxiliary distributions for lexical selectional preferences, we explore them for domain adaptation.

The idea is to exploit the information of the more general model, estimated from the larger, out-of-domain treebank, for parsing data from a particular target domain where only a small amount of training data is available. A related study is Hara et al. (2005). They also assume a limited amount of in-domain training data. However, their approach differs from ours in that they incorporate an original model as a reference distribution, and their estimation procedure is different (based on a packed parse forest rather than an informative sample, introduced in Section 2.3). Here we gauge the effect of auxiliary distributions, and later (Section 4.5.1) compare it to an approach based on reference distributions. As already mentioned, auxiliary distributions have the advantage that the contribution of the additional source is regulated through the estimated feature weight.

More specifically, we extend the target model to include (besides the original features) one additional real-valued feature (k=1).[2] Its value is defined to

[2]Or alternatively, $k \geq 1$ (cf. Section 4.4.1).

be the negative logarithm of the conditional probability given by p_{OUT}, the original, out-of-domain model. Hence, the general model is 'merged' into a single auxiliary feature:

$$f_{m+1} = -\log p_{OUT}(\omega|s) \tag{4.9}$$

The parameter of the new feature is estimated using the same estimation procedure as for the remaining model parameters. Intuitively, our auxiliary feature models dispreferences of the general model for certain parse trees. When the out-of-domain model assigns a high probability to a parse candidate, the auxiliary feature value will be small, close to zero. In contrast, a low probability parse tree in the general model gets a higher feature value. Together with the estimated feature weight expected to be negative, this has the effect that a low probability parse in the out-of-domain model will reduce the probability of a parse in the target domain.

4.2.3 An Alternative: Simple Model Combination

In this section an alternative approach is presented. Only two features are kept in the maximum entropy framework: one is the log probability assigned by the out-of-domain model, the other is the log probability assigned by the in-domain model:

$$f_1 = -\log p_{OUT}(\omega|s), f_2 = -\log p_{IN}(\omega|s)$$

Thus, the in-domain model is also collapsed into a single feature. The contribution of each feature is again scaled through feature weights λ_1, λ_2.

We can think of this as a simple instantiation of *model combination*. Alternatively, *data combination* is a baseline method where the in-domain and out-domain training data is simply concatenated and a new model is trained on the union of data (cf. the discussion in Chapter 3). A disadvantage of data combination is that the larger amount of out-domain data might 'overwhelm' the small amount of in-domain data. Instead, simple model combination interpolates the two models in a linear fashion by scaling their contribution through the estimated feature weights. Model combination is called *linear interpolation* (LinInt) in e.g. Daumé III (2007).

Note that if we skip the parameter estimation step and simply assign the two parameters equal weights, the method reduces to $p_{OUT}(\omega|s) \times p_{IN}(\omega|s)$, i.e. just multiplying the respective model probabilities.

4.3 Experimental Design

The parsing system used in this study is Alpino (introduced in Section 2.4.1), a wide-coverage natural language parser for Dutch.

To recap, Alpino consists of approximately 800 grammar rules in the tradition of HPSG and a large hand-crafted lexicon, that together with a left-corner parser constitutes the parser component. The second stage of Alpino is a statistical disambiguation component based on maximum entropy. Training the parser requires estimating parameters for the disambiguation component. The Alpino system is used to parse the training corpora, where up to $m = 1000$ derivations are recorded for every sentence.

4.3.1 Treebanks

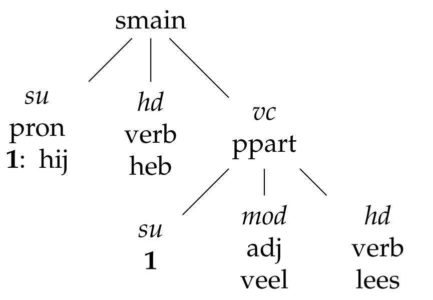
An overview of the various annotated corpora used in these experiments is given in Table 4.1. It shows size, average sentence length and ambiguity (measured as average parses per sentence).

Corpus	**Type**	**Sents**	**ASL**	**APS**
Alpino (Cdb)	newspaper text	7,136	19.7	259.28
CLEF (2003-2007)	questions	1,746	8.0	17.19
CGN	spoken data	2,101	11.1	144.05
comp-a	spontaneous conversations	1,193	8.6	80.33
comp-b	interviews with teachers	525	14.2	174.03
comp-l	commentaries (broadcast)	116	15.9	276.65
comp-m	ceremonial speeches	267	12.8	234.62

Table 4.1: Overview of the corpora. Sents = number of sentences; ASL = average sentence length; APS = average parses per sentence.

The standard version of Alpino that we use here as out-of-domain model is trained on the Alpino (Cdb) treebank (van Noord, 2006). It contains dependency structures for the Cdb (newspaper) part of the Eindhoven corpus, that contains 7,136 sentences.

For the domain-specific (target) corpora, we consider the CLEF treebank (questions; 1,746 sentences) in the first set of experiments (Section 4.4.1). In the second part (Section 4.4.2) we evaluate the approach on subdomains of the Spoken Dutch corpus (CGN, *Corpus Gesproken Nederlands* Oostdijk, 2000; spoken data; size varies). The CGN corpus contains a variety of subdomains to document various dimensions of language use (Oostdijk, 2000). We se-



⟨heb,*su*,hij⟩
⟨heb,*vc*,lees⟩
⟨lees,*su*,hij⟩
⟨lees,*mod*,veel⟩

Figure 4.1: Example of a LASSY dependency structure (Alpino output) for the sentence *hij heeft veel gelezen* 'He has read a lot' with corresponding set of dependency relations (also called dependency triples) used for evaluation. The corresponding dependency graph is shown in Figure 4.2.

lected a subset of the data from CGN to cover a variety of subdomains of different size, namely *spontaneous conversations, interviews with teachers of Dutch, commentaries/columns/reviews (broadcast), ceremonial speeches/sermons*. Furthermore, we considered only parts of the annotated data that covers standard Dutch spoken in the Netherlands (and not the southern standard Dutch, a variant spoken in Flanders, the Flemish part of Belgium). Thus, the selected subcorpora vary in size, ranging from 116 to 1,193 sentences per subdomain, indicated as comp-*x* in Table 4.1.

4.3.2 Evaluation Metrics

To evaluate the performance of a parsing system, several metrics can be employed. As the output of the parsers used in this work are dependency structures, we here focus on evaluation schemes for dependency parsing.

The output of a parser is evaluated by comparing the generated dependency structure for a sentence to the gold standard dependency structure in a treebank. For this comparison, the dependency structure (a directed acyclic graph) is represented as a set of named dependency relations (or dependency triples) of the form ⟨*head-word, dependency relation, dependent head-word*⟩. An example is given in Figure 4.1. It shows the output of Alpino (LASSY dependency structure; cf. Section 2.4.1) and the dependency triples. A more common representation of the same dependency structure is given in Figure 4.2.

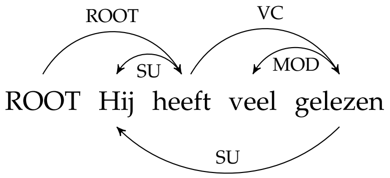


Figure 4.2: Dependency graph corresponding to the LASSY dependency structure shown in Figure 4.1. Note that the Alpino evaluation scheme does not consider the top-most relation from ROOT to the head of the sentence.

To compare such sets of dependency relations, the number of dependencies that are identical in the generated and the stored structure are counted. Let D_p^i be the number of dependencies produced by the parser for sentence i, D_g^i is the number of dependencies in the treebank parse, and D_o^i is the number of correct dependencies produced by the parser. If no superscript is used, we aggregate over all sentences of the test set, i.e.:

$$D_p = \sum_i D_p^i \qquad D_o = \sum_i D_o^i \qquad D_g = \sum_i D_g^i$$

Precision is the total number of correct dependencies returned by the parser, divided by the overall number of dependencies returned by the parser:

$$\text{precision} = D_o / D_p$$

Recall is the number of correct system dependencies divided by the total number of dependencies in the treebank:

$$\text{recall} = D_o / D_g$$

As usual, precision and recall can be combined in a single f-score metric (also known as f1-score in this particular instantiation), i.e. the harmonic mean of precision and recall:

$$\text{f-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

An alternative evaluation score for dependency structures is based on the observation that for a given sentence of n words, a parser would be expected to return exactly n dependencies, namely one dependency per word. This follows from the *single-head property* (Kübler et al., 2009). In such cases, we can simply use the percentage of correct dependencies as a measure of accuracy. This labeled (or named) dependency accuracy was used, for instance, in the CoNLL shared task on dependency parsing (Buchholz & Marsi, 2006), where the term labeled attachment score (LAS) was coined. In more detail, since $|D_p| = |D_g|$ we have:

$$\text{labeled attachment score (LAS)} = D_o / D_g$$

Instead of considering an edge correct if both the label and attachment are correct, there is also a variant of LAS that does not take the label into account, called unlabeled attachment score (UAS). It is equivalent to LAS except that a correctly attached dependency edge is still considered to be correct even though the label itself (relation name) is incorrect. Thus, UAS is usually higher than the respective LAS measure.

The evaluation metric of the Alpino parser is a variant of labeled dependency accuracy, which does allow for some discrepancy between the number of returned and expected dependencies. The resulting metric is called *concept accuracy* (CA) and is defined as:

$$\text{CA} = \frac{\sum_i D_o^i}{\sum_i \max(D_g^i, D_p^i)}$$

Such a discrepancy between returned and expected dependencies can occur, for instance, because in the syntactic annotations of Alpino words can be dependent on more than a single head (thus they do not follow the single-head property). Such 'secondary edges' (van Noord, 2006) are expressed by means of coindexation and an example is illustrated in Figure 4.1: *hij* 'he' is the subject of both the auxiliary verb *heeft* 'has' and the verb *gelezen* 'read'. Note that the standard Alpino evaluation scheme does not consider the top relation (the 'root' relation in CoNLL) for scoring. In our example, that would be the dependency triple <ROOT,*root*,heb>, i.e. the dependency edge from some artificial starting token (ROOT) to the head of the sentence *heb* (Fig. 4.2).

The concept accuracy metric can be characterized as the mean of a per-sentence minimum of recall and precision. The resulting CA score therefore is typically slightly lower than the corresponding f-score. An explanation and motivation for the use of CA over f-score is given in van Noord (2009). In this dissertation, we will report CA. Using f-score would not have altered the conclusions.

```
1: Set $c = 0$, $R =$ number of iterations
2: Compute actual statistic of score differences $|S_X - S_Y|$ on test data
3: for random shuffles $r = 0, \ldots, R$ do
4:   for sentences in test set do
5:     Shuffle variable tuples between system $X$ and $Y$ with probability 0.5
6:   end for
7:   Compute pseudo-statistic $|S_{X_r} - S_{Y_r}|$ on shuffled data
8:   if $|S_{X_r} - S_{Y_r}| \geq |S_X - S_Y|$ then
9:     $c++$
10:  end if
11: end for
12: $p = (c+1)/(R+1)$
13: Reject null hypothesis if $p$ is less than or equal to specified rejection level
```

Figure 4.3: Approximate randomization test pseudo-code.

In the experiments that follow we report concept accuracy. We compute statistical significance using the *approximate randomization test* (Noreen, 1989, chapter 2). It is a computer-intensive, non-parametric statistical hypothesis test, that has been applied to several natural language processing tasks (e.g. Yeh, 2000; Riezler & Maxwell, 2005; Cahill et al., 2008). A general introduction to computer-intensive tests can be found in Noreen (1989). A nice description of the approximate randomization test applied to natural language processing applications is given in Yeh (2000), Riezler and Maxwell (2005) and Cahill et al. (2008). The following is a summary thereof.

Computer-intensive tests (like the approximate randomization test) are designed to assess result differences of a test statistic without making assumptions on the sampling distribution of the test statistic. Thus, they are well-suited to assess significance of differences in test statistics such as f-scores. They are called computer-intensive because such tests require recomputing the test statistics for many (typically 1000 or 10000) artificially constructed data sets. The test statistics are computed by accumulating certain count variables over sentences in the test set, like the number of produced and correct dependency relations. Under the null hypothesis, the compared systems are not different, thus any variable tuple produced by one system could have been produced just as well by the other system. Thus, shuffling the variable tuples between the two systems with equal probability and recomputing the pseudo-statistic on the shuffled data creates an approximation of the test statistic distribution under the null hypothesis. For a test set of S sentences, there are 2^S possible shuffles. The approximate randomization test produces shuffles by randomization rather than requiring access to all possible assignments. The significance level is then computed by counting how often the pseudo-statistic is greater than or equal to the actual test statistic. We have implemented the approximate randomization test in R, the pseudo-code is given in Figure 4.3 (taken from Riezler & Maxwell, 2005).

4.4 Empirical Results

In the first set of experiments we focus on the question answering (QA) domain (CLEF corpus). Later we evaluate the various approaches also on subparts of the spoken Dutch corpus, *Corpus Gesproken Nederlands* (CGN). An overview of the experimental setup is given in Figure 4.4.

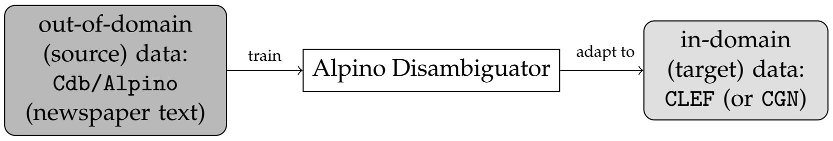

out-of-domain (source) data: Cdb/Alpino (newspaper text)
train
Alpino Disambiguator
adapt to
in-domain (target) data: CLEF (or CGN)

Figure 4.4: Supervised domain adaptation setup: source=Alpino and target=CLEF (or CGN); both are *labeled*, but there is less in-domain (target) data.

4.4.1 Experiments with the CLEF Data

Besides evaluating the auxiliary-based approach and simple model combination, we conduct several baseline experiments. An overview of the various models follows:

- In-domain (CLEF): train on CLEF (baseline); this is done using 5-fold cross validation (train on four folds, evaluate on the fifth; per-fold accuracy is reported)
- Out-domain (Alpino): train on Alpino
- Data combination (CLEF+Alpino): train a model on the union of data

- Auxiliary distribution (CLEF+Alpino_aux): add the Alpino model as auxiliary feature to CLEF
- Model combination: keep only two features $p_{OUT}(\omega|s)$ and $p_{IN}(\omega|s)$. Two variants:
    i) estimate the parameters λ_1, λ_2 (estimate λ)
    ii) give them equal weights, i.e. λ_1=λ_2=-1 (equal λ)

We assess the performance of all of these models on the CLEF data. Baselines are given in Table 4.2. To illustrate the role of the disambiguation component, Table 4.3 provides lower and upper bounds on the in-domain target data. The random model selects an arbitrary parse per sentence, while the oracle chooses the best available parse.

Data set (size: sents)	In-dom. CLEF	Out-dom. Alpino	Data combination CLEF+Alpino
CLEF 2003 (446)	97.01	94.02	<u>97.21</u>
CLEF 2004 (700)	96.60	89.88	95.14
CLEF 2005 (200)	97.65	87.98	93.62
CLEF 2006 (200)	97.06	88.92	95.16
CLEF 2007 (200)	96.20	92.48	<u>97.30</u>

Table 4.2: Baselines on the CLEF test data; underlined scores indicate results above in-domain baseline (CLEF).

CLEF	2003	2004	2005	2006	2007
Random	76.87	76.83	78.09	78.25	77.14
Oracle	99.60	99.14	99.80	99.69	99.16

Table 4.3: Lower and upper bounds of models trained on the target data.

First of all, the baselines in Table 4.2 show that, despite the smaller size of the in-domain training data, the CLEF models reach a high performance level in this question domain. Moreover, they perform significantly better than the Alpino model, trained on a larger amount of out-of-domain data. In contrast, simple data combination results in a model (CLEF+Alpino) whose performance is somewhere in between. It is able to contribute in some cases to disambiguate questions, while leading to wrong decisions in other cases.

The results of the various adaptation techniques are shown in Table 4.4. For the auxiliary-based approach (CLEF+Alpino_aux) with its regulated contribution of the general model, the results show that adding the feature does not help. On most data sets the same performance was achieved as by the in-domain model, while on only two data sets (CLEF 2005, 2007) the use of the auxiliary feature results in an insignificant improvement.

Data set	In-dom.	Aux.distribution	Model combination	
size (sents)	CLEF	CLEF+Alpino_aux	estimate	equal
CLEF 2003 (446)	97.01	97.01	<u>97.14</u>	<u>97.46</u> ⋆
CLEF 2004 (700)	96.60	96.60	<u>97.12</u> ⋆	<u>97.23</u> ⋆
CLEF 2005 (200)	97.65	<u>97.72</u>	<u>97.99</u>	<u>98.19</u>
CLEF 2006 (200)	97.06	97.06	97.00	96.45
CLEF 2007 (200)	96.20	<u>96.33</u>	<u>96.33</u>	<u>96.46</u>

Table 4.4: Results on the CLEF test data; underlined scores indicate results above in-domain baseline (CLEF). Entries marked with ⋆ are statistically significant at $p < 0.05$ with respect to the in-domain baseline.

In contrast, simple model combination works surprisingly well. On almost all data sets model combination yields an improvement over all other models. On only one data set (CLEF 2006) it falls slightly below the in-domain baseline, but still considerably outperforms data combination (Table 4.2). This is true for both model combination methods, those with estimated and those with equal weights. In general, the results show that model combination outperforms data combination (with the exception of one data set, CLEF 2007), and the simplest model combination technique with equal weights often performs best (e.g. on two test sets it reaches a significant improvement over the in-domain baseline).

Contrary to expectations, the auxiliary-based approach performs poorly and often could not even come close to the results obtained by simple model combination. In the following we explore possible reasons for this result.

Examining Possible Causes

Is the feature ignored? One possible point of failure could be that the auxiliary feature is simply ignored. If the estimated weight had been close to zero the feature would indeed not contribute to the disambiguation task. Therefore, we examined the estimated weights for that feature. From that analysis we saw that, compared to the other features, the auxiliary feature got a weight relatively far from zero. It got on average a weight of -0.0905 in our data sets and as such is among the most influential weights, suggesting it to be important for disambiguation.

Is the feature properly modeling the out-of-domain model? Another question that needs to be asked is whether the auxiliary feature is properly modeling the original Alpino model. For this sanity check, we create a model that contains only the single auxiliary feature and no other features. The weight of the feature is set to a constant negative value. Alternatively, we may estimate its weight, but as it does not have competing features we are safe to assume it constant. The performance of the model is assessed on the CLEF test data. The result in Table 4.5 shows that the auxiliary feature is indeed properly modeling the general Alpino model, as the two models result in identical performance.

Data set	only aux	Alpino
CLEF2003	94.02	94.02
CLEF2004	89.88	89.88
CLEF2005	87.98	87.98
CLEF2006	88.92	88.92
CLEF2007	92.48	92.48

Table 4.5: Sanity check: a model that only contains the auxiliary feature versus out-of-domain Alpino model.

Is the single feature too weak? In the experiments so far the general model was 'packed' into a single feature value. To check whether the auxiliary feature alone is too weak, we examine the inclusion of several auxiliary distributions ($k > 1$). Each auxiliary feature that is added represents a *submodel* corresponding to an actual feature template class used in the original model. The value of the feature is the negative log-probability as defined in equation (4.9), where OUT corresponds to the respective Alpino submodel.

The current disambiguation model of Alpino uses 21 feature templates (introduced in Section 2.4.1). From these feature templates, we create two models that vary in the number of classes used. In the first model (5 class), five

($k = 5$) auxiliary distributions are used which correspond to five clusters of feature templates. They are defined manually and correspond to submodels for part-of-speech, dependencies, grammar rule applications, bilexical preferences and the remaining Alpino features. In the second model (21 class), every single feature template is simply used as its own cluster ($k = 21$).

We test the two models and compare them to our baseline (the model trained on in-domain CLEF data). The results of these experiments are given in Table 4.6. They show that both the 5 class and the 21 class model do not achieve any improvement over the in-domain baseline (CLEF), nor over the single auxiliary model (CLEF+Alpino_aux).

Data set (sents)	CLEF	CLEF+Alpino_aux	5 class	21 class
CLEF2003 (446)	97.01	97.01	97.01	97.04
CLEF2004 (700)	96.60	96.60	96.57	96.60
CLEF2005 (200)	97.65	97.72	97.72	97.72
CLEF2006 (200)	97.06	97.06	97.06	97.06
CLEF2007 (200)	96.20	96.33	96.20	96.27

Table 4.6: Results on CLEF including several auxiliary features corresponding to Alpino submodels versus in-domain CLEF baseline and single auxiliary distribution approach.

What if we have smaller amounts of in-domain training data? Our expectation is that the auxiliary feature is at least helpful in the case very little in-domain training data is available. Therefore, we evaluate the approach with smaller amounts of training data.

We sample (without replacement) a specific number of training instances from the original CLEF data and train models on the reduced training data. The resulting models are tested with and without the additional feature as well as model combination. Table 4.7 reports the results of these experiments for models trained on a portion of up to 5% CLEF data. Figure 4.5 illustrates the overall change in performance of models trained on up to 60% of the data (the figure shows the learning curve for the CLEF 2004 data set - similar figures result from the other CLEF data sets).

Obviously, an increasing amount of in-domain training data improves the accuracy of the models. However, for the auxiliary feature, the results in Table 4.7 show that the models with and without the auxiliary feature result in an almost identical performance (therefore we depict only one of the lines in

	0%	1% CLEF				5% CLEF			
CLEF	Alpino	In-dom.	+aux	MC_λ	$MC_=$	In-dom.	+aux	MC_λ	$MC_=$
2003	94.02	91.93 (4)	91.93	95.59	93.65	93.83 (22)	93.83	95.74	95.17
2004	89.88	86.59 (7)	86.59	90.97	91.06	93.62 (35)	93.62	93.42	92.95
2005	87.98	87.34 (2)	87.41	91.35	89.15	95.90 (10)	95.90	97.92	97.52
2006	88.92	89.64 (2)	89.64	92.16	91.17	92.77 (10)	92.77	94.98	94.55
2007	92.48	91.07 (2)	91.13	95.44	93.32	94.60 (10)	94.60	95.63	95.69

Table 4.7: Results on the CLEF data with varying amount of domain-specific training data (number of sentences in parentheses). First column: standard Alpino performance on the data set (0% CLEF/Alpino); in-dom: performance of model trained on certain percentage of in-domain training data; +aux: in-domain model augmented with auxiliary feature that represents the general, out-of-domain Alpino model; MC: model combination (in particular, MC_λ with estimated weights and $MC_=$ with equal/constant weights). Underlined scores represent performance levels above in-domain baseline.

Figure 4.5, the auxiliary distribution line). Hence, the inclusion of the auxiliary feature does not help in this case either. The models achieve similar performance independently of the available amount of in-domain training data.

In more detail, the auxiliary-based approach does not perform as well as model combination, but usually works better than simply including all data (data combination; cf. Fig. 4.5). However, in contrast to those techniques, the auxiliary-based approach even hurts performance: on models trained on very little in-domain training data (e.g. 1% CLEF training data, which corresponds to on average approximately three questions), the performance falls even *below* the original Alpino model accuracy (0% column in Table 4.7, i.e. straight dashed line in Fig. 4.5). Thus, including the out-of-domain model through the auxiliary feature does not help.

In contrast, simple model combination is more beneficial. It is able to outperform almost constantly the in-domain baseline (CLEF) and the auxiliary-based approach (CLEF+Alpino_aux). Furthermore, in contrast to the latter, model combination almost never falls below the out-of-domain (Alpino) baseline (except slightly on the CLEF 2003 data set with 1% training data). And, as shown in Figure 4.5, even 30% of the training data (on that particular data set) is enough for model combination to reach the performance level of a model

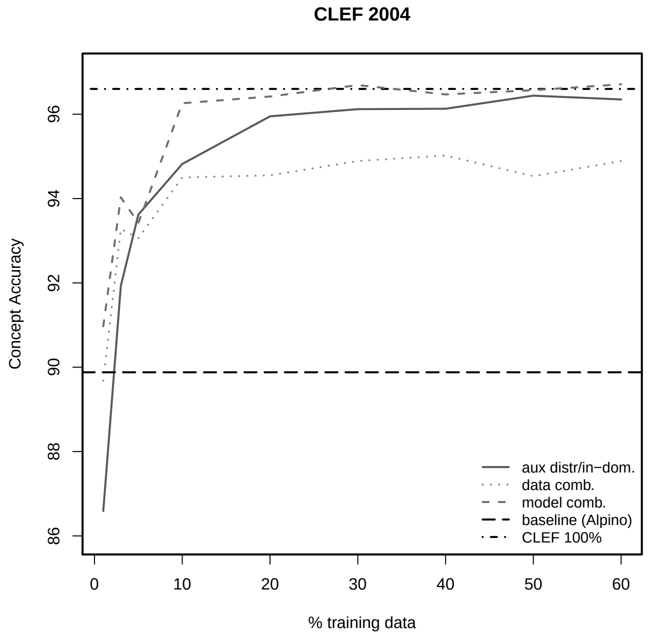


Figure 4.5: Amount of in-domain training data versus accuracy on CLEF 2004 (similar figures result from the other CLEF data sets) - note that since the performance of the auxiliary distribution approach is nearly indistinguishable from the in-domain baseline (cf. Table 4.7) we only depict one of the lines. The auxiliary-based approach does not work as well as model combination; but it usually works better than simply including the OUT data (data combination).

trained on all (100%) of the in-domain data.

We would have expected the auxiliary feature to be useful at least when very little in-domain training data is available. However, the empirical results reveal the contrary. We tried to rule out several possible causes for this negative result. Amongst others, the suspicion that the (non-auxiliary) features may overwhelm the single auxiliary feature, so that possible improvements by increasing the feature space on this small scale might be invisible. We believe this is not the case. Other studies have shown that including just a few features might indeed help (Johnson & Riezler, 2000; van Noord, 2007) (e.g. the former just added three features).

We believe that the reason for this drop in performance is the amount of available in-domain training data and the corresponding scaling of the weight of the auxiliary feature. When little training data is available, the weight cannot be estimated reliably and hence is not contributing enough compared to the other features (exemplified in the drop of performance from 0% to 1% training data in Table 4.7). In such cases it is more beneficial to just apply the original Alpino model or the simple model combination technique.

4.4.2 Experiments with CGN

One might argue that the question domain is rather 'easy', given the high baseline performance and the fact that a few hand-annotated questions are enough to obtain a reasonable model. Therefore, we examine the auxiliary-based approach on CGN subdomains (*Corpus Gesproken Nederlands*), the Spoken Dutch Corpus (Oostdijk, 2000).

The empirical results per CGN subdomain are given in Table 4.8. The table shows the accuracy of models trained on the relevant subcorpora (In-dom.), the result of Alpino and the performance of the two adaptation approaches, auxiliary distribution and model combination. In contrast to the question domain, the general accuracy level is much lower on this more heterogeneous, spoken data.

The results show that the auxiliary-based approach does not work on these CGN subdomains either. It just performs about as well as the in-domain model, trained on the respective corpus. Even on subdomains where Alpino clearly outperforms the in-domain baseline (like comp-l, *broadcast commentaries*, or comp-m, *ceremonial speeches*), the auxiliary feature does not contribute to improved parsing performance. Moreover, the auxiliary-based approach is not able to improve accuracy on data sets where very little training data is available (e.g. comp-l), thus confirming our previous finding. In some cases the auxiliary feature rather, although only slightly, *degrades* performance (indicated in italic in Table 4.8) and performs worse than the counterpart model without the additional feature.

Depending on the characteristics of data/domain and its size, the best model adaptation method varies on CGN. On some subdomains simple model combination performs best, while on others it is more beneficial to just apply the original, out-of-domain Alpino model. To conclude, model combination achieves in most cases a modest improvement, while we have shown em-

Data set	In-dom.	Aux.Distr. (+aux)	Out-dom. Alpino	Model combination estimate	Model combination equal
comp-a (1,193) - Spontaneous conversations ('face-to-face')					
fn000250	63.20	63.28	*62.90*	63.91	**63.99**
fn000252	64.74	64.74	*64.06*	64.87	**64.96**
fn000254	66.03	*66.00*	*65.78*	66.39	**66.44**
comp-b (525) - Interviews with teachers of Dutch					
fn000081	66.20	66.39	66.45	**67.26**	66.85
fn000089	62.41	62.41	63.88	**64.35**	64.01
fn000086	62.60	62.76	63.17	63.59	**63.77**
comp-l (116) - Commentaries/columns/reviews (broadcast)					
fn000002	67.63	67.63	**77.30**	76.96	72.40
fn000017	64.51	*64.33*	**66.42**	66.30	65.74
fn000021	61.54	61.54	**64.30**	64.10	63.24
comp-m (267) - Ceremonial speeches/sermons					
fn000271	59.25	59.25	63.78	**64.94**	61.76
fn000298	70.33	*70.19*	74.55	**74.83**	72.70
fn000781	72.26	72.37	**73.55**	**73.55**	73.04

Table 4.8: Excerpt of results on various CGN subdomains (number of sentences per component in parentheses). In-dom.: performance of a parser trained on relevant subdomain; Aux.Distr. (+aux): adding Alpino model as auxiliary feature to in-domain model. Best performance in boldface; performance below in-domain baseline in italic, above baseline is underlined.

pirically that the domain adaptation method based on auxiliary distributions performs about as well as a model trained on in-domain data.

4.5 Comparison to Prior Work

In this section, we compare the auxiliary-based approach to two previously suggested supervised domain adaptation techniques: the *reference distribution* approach (Hara et al., 2005) and *easy adapt* (Daumé III, 2007).

4.5.1 Reference Distribution

The idea is similar to the approach based on auxiliary distributions: exploit an original, out-of-domain model that was trained on a larger treebank as additional information source. However, rather than incorporating it as additional log-weighted features, the distribution is integrated as *reference distribution*. That is, instead of estimating parameters for the maximum entropy model so that the resulting model minimally diverges from a uniform distribution this uniform distribution is now replaced by another, hopefully more informative, reference distribution $q_0(\omega|s)$ (previously denoted as $q(\omega|s)$). The resulting model is also known as maximum entropy minimum divergence (MEMD) model.

$$p(\omega|s) = \frac{1}{Z(s)} q_0(\omega|s) \exp(\sum_{i=1}^{m} \lambda_i f_i(\omega, s)) \tag{4.10}$$

This was the approach adopted by Hara et al. (2005). They integrated an original, out-of-domain model estimated from the Penn Treebank (newspaper text) into a model trained on biomedical abstracts (Genia corpus), thereby obtaining an improvement over both the in-domain baseline and simple data combination (cf. Table 3.3 – to summarize, their model reached an f-score of 87.87 compared to data combination 86.32 and in-domain baseline 85.72).

We tested the reference distribution approach on our CLEF data. That is, we considered $p_{OUT}(\omega|s)$ (the out-of-domain Alpino model) as reference distribution when estimating parameters for the CLEF models.

Data set size (sents)	In-dom. CLEF	Aux.distr. CLEF+Alp_aux	Ref.distr.
CLEF 2003 (446)	97.01	97.01	82.37
CLEF 2004 (700)	96.60	96.60	80.32
CLEF 2005 (200)	97.65	97.72	81.57
CLEF 2006 (200)	97.06	97.06	81.00
CLEF 2007 (200)	96.20	96.33	78.43

Table 4.9: Results of using the source model as reference distribution q_0.

The results in Table 4.9 show that the reference distribution approach does not work: the model results in the overall lowest performance, scoring below the in-domain baseline but even below the out-of-domain (Alpino) baseline.

This suggests that relying too much on the external source (original distribution) has a deteriorating effect on performance.

Compared to the reference distribution, the auxiliary-based approach that scales the contribution of the original model through the estimated feature weight would be a viable approach overall, despite its empirically shown weaknesses.

4.5.2 Easy Adapt

In this section, we report on applying *easy adapt* (Daumé III, 2007), introduced in section 3.4. The training data is generated as follows: take each feature in the source and target domain, duplicate it in order to create source-specific and general, as well as target-specific and general versions (three in total). Train a model on the union of data, augmented with the new features. In this way, the training data contains source-specific, target-specific and general features. Intuitively, through this simple transformation of the feature space, the supervised learning algorithm is supposed to 'learn' which features transfer better to the new domain. We evaluate the approach on both application domains, CLEF and CGN. The results are given in Table 4.10 and 4.11, respectively.

Data set size (sents)	In-dom. CLEF	Out-dom. Alpino	Easy adapt
CLEF 2003 (446)	97.01	94.02	96.99
CLEF 2004 (700)	96.60	89.88	94.88
CLEF 2005 (200)	97.65	87.98	93.15
CLEF 2006 (200)	97.06	88.92	96.02
CLEF 2007 (200)	96.20	92.48	96.01

Table 4.10: Results on the CLEF test data including easy adapt results.

Table 4.10 shows that the performance of easy adapt falls below the in-domain baseline on all CLEF test sets. On some data sets, easy adapt has a substantial deteriorating effect. A possible explanation for this might be that the in-domain baseline (CLEF) not only does *slightly* better (but rather performs substantially better) than the out-domain model. That is, we do not meet the requirement of easy adapt, which is appropriate "exactly in the case when one has enough target data to do slightly better than just using only source data" (Daumé III, 2007). Our empirical results suggest that on this question domain a small amount of annotated in-domain data is already enough to obtain a model of reasonable performance.

Data set	In-dom.	Out-dom.	Easy adapt
comp-a (1,193) - Spontaneous conversations			
fn000250	63.20	*62.90*	**63.67**
fn000252	64.74	*64.06*	**64.85**
fn000254	66.03	*65.78*	**66.22**
comp-b (525) - Teacher interviews			
fn000081	66.20	**66.45**	*65.95*
fn000089	62.41	<u>63.88</u>	**64.08**
fn000086	62.60	**63.17**	<u>62.69</u>
comp-l (116) - Commentaries (broadcast)			
fn000002	67.63	**77.30**	<u>72.06</u>
fn000017	64.51	**66.42**	*64.07*
fn000021	61.54	**64.30**	*60.28*
comp-m (267) - Ceremonial speeches			
fn000271	59.25	**63.78**	<u>60.30</u>
fn000298	70.33	**74.55**	<u>72.98</u>
fn000781	72.26	**73.55**	*71.60*

Table 4.11: Easy adapt results on CGN components.

Table 4.11 reports the results on the more heterogeneous CGN data. On the subdomain *spontaneous conversations* (comp-a), where we do meet the precondition, easy adapt is able to improve the in-domain baseline slightly. Furthermore, what is consistent with previous findings (Daumé III, 2007) is that in cases where the out-of-domain model outperforms the domain-specific in-domain model (such as on CGN domains *teacher interviews*, *commentaries* and *ceremonial speeches*) easy adapt will fail. Daumé III (2007) believes the reason for this is that source and target domains are not that different: "If the domains are so similar that a large amount of source data outperforms a small amount of target data, then it is unlikely that blowing up the feature space will help." We do not believe this to be the case here, as, for instance, *ceremonial speeches* (comp-m) and newspaper text do not seem to be very similar domains either, nor do we suppose this to be the only reason for the approach to fail.

Factors such as training set size should also play a considerable role, but this would need further investigation. However, a question which emerges here is: How can we measure domain similarity? We will return to this issue in later chapters.

We conclude that easy adapt does not really work in our case. Often, either just applying the Alpino model or simple model combination was more beneficial.

4.6 Summary and Conclusions

In this chapter we focused on supervised domain adaptation techniques to adapt a discriminative parse selection model to novel domains. We examined auxiliary distributions for domain adaptation, and compared them to a range of straightforward baselines (data combination, model combination) and prior work (reference distribution, easy adapt).

While the auxiliary approach has been successfully applied to learn lexical selectional preferences (Johnson & Riezler, 2000; van Noord, 2007), the empirical results show that integrating a more general model into a domain-specific model through the auxiliary feature(s) does not help. The auxiliary approach needs training data to estimate the weight(s) of the auxiliary feature(s). When little in-domain training data is available, the weight cannot be estimated appropriately and hence does not contribute enough compared to the other features. The resulting models usually just achieve in-domain baseline performance. This result was confirmed on both examined domains, questions and spoken data. In contrast, both prior-work techniques fell below the in-domain baseline; the reference distribution turned out to be the worst performing technique. This suggests that relying too much on the out-of-domain model has a deteriorating effect, thus favoring the scaling property of the auxiliary distribution, despite its shown weaknesses.

We conclude that the auxiliary feature approach is not appropriate for integrating information from a more general model to leverage limited in-domain data. Better results were achieved either without adaptation or by simple model combination.

In the next chapter, we will investigate other possibilities for parser adaptation, especially techniques for *unsupervised* domain adaptation, which assume that there is no labeled in-domain data available.

Chapter 5

Unsupervised Domain Adaptation

In this chapter we evaluate the two main approaches to unsupervised domain adaptation that can be found in the literature. We apply them to adapt a syntactic disambiguation model to new domains. More specifically, we examine the bootstrapping approach of *self-training* and compare it to the more involved technique called *structural correspondence learning* (SCL).[1]

5.1 Introduction and Motivation

As discussed in the previous chapter, studies on supervised domain adaptation have shown that straightforward baselines (e.g. models trained on source only, target only, or the union of the data) achieve relatively high performance levels and are "surprisingly difficult to beat" (Daumé III, 2007). Thus, one conclusion from that line of work is that as soon as there is a reasonable (sometimes even small) amount of labeled target data, it is often more fruitful to either just use that or to apply simple adaptation techniques (like model combination, cf. Chapter 4).

In contrast, *unsupervised domain adaptation*[2] – the scenario in which there are no annotated resources available for the new domain – is a much more realistic situation, but is clearly also considerably more difficult. While sev-

[1]Preliminary results of this chapter were published in Plank (2009b) and Plank (2009a).

[2]As discussed in Chapter 3, what we call unsupervised domain adaptation here was (before 2010) often referred to as semi-supervised DA (Daumé III, 2007). However, since the emergence of techniques that assume both labeled and unlabeled data from the target domain (which are now referred to as semi-supervised DA) there has been a shift in naming. We follow Daumé III et al. (2010) and Chang et al. (2010) and refer to unsupervised DA as the scenario in which only unlabeled target data is available.

eral authors have looked at the supervised adaptation case (e.g. Gildea, 2001; Roark & Bacchiani, 2003; Hara et al., 2005; Chelba & Acero, 2006; Daumé III, 2007; Finkel & Manning, 2009), there are fewer – and especially less successful – studies on unsupervised domain adaptation (McClosky et al., 2006; Blitzer et al., 2006; Dredze et al., 2007). Current studies on unsupervised domain adaptation show rather mixed results.

For instance, McClosky et al. (2006) show that together with a second-stage parse reranker the performance of a statistical PCFG parser can be improved by *self-training*. Previously it was generally assumed that self-training does not help. Another technique that uses unlabeled data is *structural correspondence learning* (SCL), introduced by Blitzer et al. (2006). They have shown its effectiveness on two tasks, PoS tagging and sentiment classification (Blitzer et al., 2006, 2007). In the CoNLL 2007 shared task on domain adaptation there was an attempt to apply SCL to adapt a dependency parser (Shimizu & Nakagawa, 2007). The system just ended up at place 7 out of 8 teams. However, their results are inconclusive due to a bug in their system and the general problem of annotation differences in the data sets (Dredze et al., 2007).[3] In fact, Dredze et al. (2007) report on "frustrating" results on the CoNLL 2007 shared task on domain adaptation for parsing, where the goal was to exploit unlabeled data, i.e. "no team was able to improve target domain performance substantially over a state-of-the-art baseline". Thus, the effectiveness of SCL remains rather unexplored for parsing.

Therefore, in this chapter we examine structural correspondence learning for domain adaptation of a stochastic parse selection component and compare it to various instantiations of self-training. The parsing system that we will use is the same as in the previous chapter, namely, the Alpino parser (introduced in Section 2.4.1). For the empirical evaluations, we will explore Wikipedia as primary test and training collection.

The remaining part of this chapter is structured as follows. First, we review self-training. Then, we introduce structural correspondence learning and depict our application of SCL to parse disambiguation. In Section 5.3 we present the data sets and describe the process of constructing target domain data from Wikipedia. Section 5.4 presents and discusses the empirical results.

[3]As shown in Dredze et al. (2007) and already mentioned in Chapter 3, the biggest problem with the shared task was that the data provided were annotated with different annotation guidelines, thus the general impression was that the task was ill-defined (Nobuyuki Shimizu, personal communication).

5.2 Exploiting Unlabeled Data

A general machine learning approach that uses unlabeled data is *bootstrapping*, which includes *self-training*. In contrast, *structural correspondence learning* is a technique specific to domain adaptation that tries to find a common feature representation between domains to implicitly link domain-specific features. Next, we will introduce self-training and its variants, and then we will describe structural correspondence learning.

5.2.1 Self-training

Self-training is a general single-view bootstrapping algorithm. A model is trained on labeled ('seed') data and then used to label a pool of unlabeled data. The automatically labeled data is then taken at face value and combined with the original labeled data to train a new model. This process can be iterated.

When self-training is applied to domain adaptation, the only difference to the general machine learning setup is that the labeled and unlabeled data are from different domains: the labeled data is from the *source* domain, while all unlabeled data comes from the *target* domain. This is illustrated in Figure 5.1.

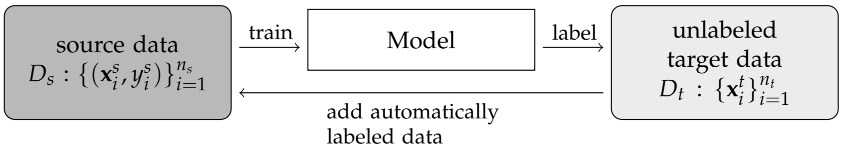

source data
$D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
train
Model
label
unlabeled
target data
$D_t : \{\mathbf{x}_i^t\}_{i=1}^{n_t}$
add automatically
labeled data

Figure 5.1: Self-training for domain adaptation: A model is trained on the labeled source data and used to annotate unlabeled target data. The automatically labeled target data is then added to the source data and a new model is trained. The process can be iterated.

There are many possible ways to instantiate self-training (Abney, 2007). For instance, whether to run a single iteration or multiple iterations of self-training; whether to select parts of the automatically labeled data or add it all, and how to select data, correspondingly. In the following, we will examine the following variants of self-training:

- Single versus multiple iterations.
- Selection versus no selection (taking all automatically labeled data or selecting presumably higher quality training instances).

For selection, we examine three simple scoring functions, where s stands for a sentence, $\Omega(s)$ is the set of parses for that sentence, and ω is a parse from $\Omega(s)$:

- *Sentence length*: $|s|$
- *Number of parses*: $|\Omega(s)|$
- *Entropy*: $-\sum_{\omega\in\Omega(s)} p(\omega|s)\log p(\omega|s)$

All scoring functions are based on the intuition that the parser should be more confident on presumably 'easier' examples. Therefore, we will select shorter sentences, sentences that contain fewer parses or have lower entropy.

So far, studies on self-training have mainly focused on generative constituency parsing (Steedman et al., 2003; McClosky et al., 2006; Reichart & Rappoport, 2007). Steedman et al. (2003) as well as Reichart and Rappoport (2007) examine self-training in the small seed case (with fewer than $1,000$ labeled sentences). While self-training with a small seed was assumed not to work (Steedman et al., 2003) (depending on the parser it either just gave a minor improvement or actually hurt performance), Reichart and Rappoport (2007) have shown that self-training can help parser performance in the small seed case when all the automatically labeled data is added to the seed data. In contrast, McClosky et al. (2006) focus on the large seed scenario and exploit a parser with reranker. Improvements are obtained (McClosky et al., 2006; McClosky & Charniak, 2008), showing that a reranker is necessary for successful self-training in such a large-seed scenario. While they all apply self-training to a generative model, we examine self-training for the adaptation of a discriminative parse selection system. We will compare it to structural correspondence learning, introduced next.

5.2.2 Structural Correspondence Learning

Structural correspondence learning (SCL) is a domain adaptation algorithm for feature-based classifiers proposed by Blitzer et al. (2006). It is inspired by structural learning (Ando & Zhang, 2005), which is a general semi-supervised machine learning algorithm. Structural correspondence learning exploits unlabeled data from both source and target domains to learn a representation under which features from different domains are assumed to be aligned. Once such correspondences are induced, they are incorporated in the labeled source data as new features. A new classifier is then trained on the augmented source data, with the goal to "convert an effective source model into an effective target model" (Blitzer, 2008). Intuitively, it is a way to obtain weights for features that otherwise had not been observed. As we will see, the correspondences are rather implicit in SCL.

$feat_X$	$\leftrightarrow$	**pivot feature**	$\leftrightarrow$	$feat_Y$
domain A		("linking" feature)		*domain B*
"boring"		"don't buy"		"defective"

Figure 5.2: Structural correspondence learning relies on *pivot features* that link features from different domains.

Before describing the algorithm in detail, let us illustrate the intuition of SCL with an example borrowed from Blitzer et al. (2007). Suppose we have a sentiment analysis system trained on book reviews (domain A), and we would like to adapt it to work well for classifying reviews of kitchen appliances (domain B). Features such as "boring" and "repetitive" are common ways to express negative sentiment in A, while "not working" or "defective" are specific to B. If there are features across domains, e.g. "don't buy" (cf. Figure 5.2), with which the domain-specific features are highly correlated, then we might tentatively align those features. Intuitively, if we are able to find good correspondences between features from different domains, then the labeled source domain data should help to generalize better to a new target domain, for which no labeled data is available.

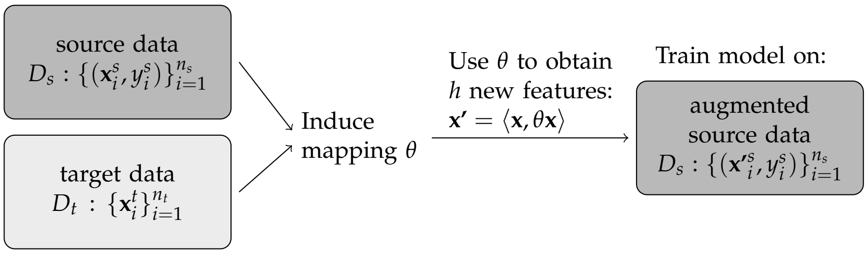

source data
$D_s : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
target data
$D_t : \{\mathbf{x}_i^t\}_{i=1}^{n_t}$
Induce mapping θ
Use θ to obtain h new features: $\mathbf{x'} = \langle \mathbf{x}, \theta\mathbf{x} \rangle$
Train model on:
augmented source data
$D_s : \{(\mathbf{x'}_i^s, y_i^s)\}_{i=1}^{n_s}$

Figure 5.3: The SCL algorithm exploits data from both source and target domain to induce correspondences between features from the two domains, encoded in the mapping θ. Then, new features are added to the source data by applying the projection $\theta\mathbf{x}$, and a new model is trained on the augmented source data (in this way weights for the original features, $\mathbf{w}$, and for the new features, $\mathbf{v}$, are estimated, cf. equation 5.2).

The key idea of SCL is to automatically identify correspondences between features from different domains by modeling their correlation with so called *pivot features*. Pivots should be features that occur frequently and behave similarly in the two domains (Blitzer et al., 2006). They are similar to auxiliary problems in structural learning (Ando & Zhang, 2005). Non-pivot features that correspond with many of the same pivot-features are assumed to correspond. These correspondences are encoded in the projection matrix θ.

```
Require: labeled source data $D_{s,l} : \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_{s,l}}$, unlabeled source $D_{s,u} :$
    $\{\mathbf{x}_i^s\}_{i=1}^{n_{s,u}}$ and target data $D_t : \{\mathbf{x}_i^t\}_{i=1}^{n_t}$
1: Select $p$ pivot features from the unlabeled data $D_{s,u} \cup D_t$
2: Train $p$ binary classifiers (pivot predictors)
3: Create matrix $W_{q \times p}$ of binary predictor weight vectors $W = [\mathbf{w}_1, .., \mathbf{w}_p]$,
   where $q$ is the number of non-pivot features
4: Apply SVD to $W$: $W_{q \times p} = U_{q \times q} D_{q \times p} V_{p \times p}^T$ where $\theta = U_{[1:h,:]}^T$ are the $h$ top
   left singular vectors of $W$
5: Apply projection $\theta \mathbf{x}$ to the source data instances $\mathbf{x} \in D_{s,l}$ and train a new
   model on the augmented source data
```

Figure 5.4: Structural correspondence learning algorithm.

The outline of the algorithm is given in Figure 5.4. Figure 5.3 depicts the main steps of the algorithm graphically. The first step of structural correspondence learning is to identify p pivot features from the set of all features in the source and target domains. Pivot features should be frequent in both domains and should align well with the task at hand. In practice, some hundreds of pivot features will be selected. All remaining non-selected features are so called non-pivot features. We denote the number of pivot features by p and q is the number of non-pivot features.

The next step (line 2 in Figure 5.4) is to train pivot predictors, which are classifiers that predict a pivot using all non-pivots. Thus, a binary classifier of the form: "Does the pivot feature occur in this instance?" is trained for each pivot. The training data for the classifiers is obtained by masking the pivot in the unlabeled data from both domains. After training, we get a weight vector $\mathbf{w}_i$ of length q (the number of non-pivots) for each pivot, i.e. $1 \leq i \leq p$. Positive entries in the weight vector indicate that a non-pivot is helpful for predicting the respective pivot feature. Non-pivots with similar weights are assumed to behave similarly and thus to correspond. The algorithm "uses co-occurrence between pivot and non-pivot features to learn a representation under which features from different domains are aligned" (Blitzer, 2008).

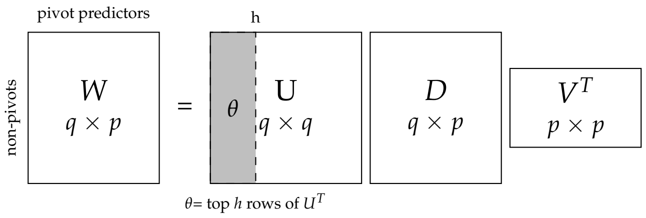


Figure 5.5: Singular value decomposition of matrix W (whose columns correspond to the weight vectors of the pivot predictors). The top h rows of U^T (the top left singular vectors; gray area) form the projection matrix θ.

Step 3 is to arrange the p weight vectors in a matrix $W_{q \times p}$ (cf. the left matrix W in Figure 5.5), where a column corresponds to a pivot predictor weight vector. Applying the projection $W^T\mathbf{x}$ (where $\mathbf{x}$ is a training instance) would give us p new features, however, for "both computational and statistical reasons" (Blitzer et al., 2006) a low-dimensional approximation of the original feature space is computed by applying a *singular value decomposition* (SVD) on W (step 4, illustrated in Figure 5.5). Let $\theta_{h \times q} = U^T_{[1:h,:]}$ be the top h left singular vectors of W (where h is a parameter specifying the dimension of θ and q is the number of non-pivot features). The resulting θ is a projection onto a lower dimensional space $\mathbb{R}^h$, parameterized by h. This projection matrix θ implicitly contains the induced feature correspondences.

The final step of SCL is to train a linear predictor on the augmented labeled source data $\mathbf{x'} = \langle \mathbf{x}, \theta\mathbf{x} \rangle$. In more detail, the original feature vector $\mathbf{x}$ is augmented with h new features obtained by applying the projection $\theta\mathbf{x}$. We obtain h new features since θ is of size $h \times q$ and an $\mathbf{x}$ is of size $q \times 1$.

$$\theta \cdot \mathbf{x} = h \text{ new features} \tag{5.1}$$

Thus h new features are added to the source domain data, and a new model is estimated. In this way, weights for the original source domain features, $\mathbf{w}$, and the h new features $\mathbf{v}$ are obtained:

$$\{\mathbf{xw} + (\theta\mathbf{x})\mathbf{v}\} \tag{5.2}$$

If θ contains meaningful correspondences, then the predictor trained on the augmented data should transfer well to the new domain. Note that a single new feature is the sum of the projection of a row of θ onto a training instance $\mathbf{x}$. This means that the feature correspondences are used in a rather implicit way – it might be rather hard to trace back which feature correspondences were the most helpful.

SCL for Parse Selection

This section describes our application of structural correspondence learning to the parse disambiguation model of the Alpino parser. We will introduce and discuss all our design choices.

Pivot Features A property of the pivot predictors is that they can be trained from unlabeled data, as they represent properties of the input. So far, pivot features on the word level were used (Blitzer et al., 2006, 2007; Blitzer, 2008), e.g. "Does the bigram *not buy* occur in this document?" (Blitzer, 2008). Pivot features are the key ingredient for SCL, and they should align well with the NLP task at hand. For PoS tagging and sentiment analysis, features on the word level are intuitively well-related to the task. For parse disambiguation based on a conditional model this is not the case, as words alone are irrelevant. They are constant within a parse disambiguation context and thus do not help in choosing a parse out of the possible analyses.

Therefore, we actually introduce an additional and new layer of abstraction, which, we hypothesize, aligns well with the task of parse disambiguation: we first *parse* the unlabeled data with the grammar. In this way we obtain full parses for given sentences, allowing access to more abstract representations of the underlying pivot predictor training data, even though the data might be noisy. Thus, instead of using word level features, features in our scenario correspond to properties of the generated parses. The features are those described in Chapter 2 (page 30) and encode information about, for instance, the application of grammar rules (*r1,r2* features), dependency relations (*dep*), PoS tags (*f1,f2*), bilexical preferences (*z*), appositions (*appos*) and further syntactic features (e.g. for unknown words and coordination).

Selection of Pivot Features and Removal of Predictive Features As pivot features should be common across domains and should align well with the task at hand, we restrict our pivots to be of the following type: grammar rule applications (the *r1* features). We count how often each feature appears in the parsed source and target domain data, and select those *r1* features as *pivot* *features*, whose count is $> t$, where t is a specified threshold. In all our experiments, we set $t = 200$. In this way, we obtained on average 120 pivot features, on the data sets that will be described in Section 5.3.2.

As pointed out by Blitzer et al. (2006), each training instance for the pivot predictor will actually contain features which are totally predictive for the pivot (i.e. the pivot itself). In our case, we additionally have to pay attention to 'more specific' features, namely the *r2* features. The *r2* features extend the *r1* features (that represent the application of a grammar rule) since the *r2* features incorporate more information than their *r1* parent (i.e. which grammar rules applied in the construction of daughter nodes, see Chapter 2, page 30 and following, for an example). It is crucial to remove these predictive features when creating the training data for the pivot predictors.

To train the pivot predictors, we extract the most probable parse per sentence according to the source domain model from the concatenation of the source and automatically labeled target data. This has been done for practical reasons to limit the size of the training data for the pivot predictors. For every such training instance, we mask the pivot in the data and train a binary classifier using the remaining non-predictive nonpivot features. This gives a pivot predictor weight vector for every pivot feature, which form the columns of the matrix W (Figure 5.5). In an earlier study (Plank, 2009b) we followed Blitzer et al. (2006), who in turn followed Ando and Zhang (2005), and we only kept the positive entries in the pivot predictors weight vectors to compute the SVD. Thus, when constructing the matrix W, we disregarded all negative entries in W and compute the SVD ($W = UDV^T$) on the resulting non-negative sparse matrix. This sparse representation saves both time and space. However, in the experiments reported in this chapter we keep all entries as we found that it gives slightly better results.

Further General Practical Issues of SCL

In practice, there are more free parameters and model choices besides the ones discussed above (Ando & Zhang, 2005; Blitzer et al., 2006; Blitzer, 2008). We will introduce them next.

Feature Normalization and Feature Scaling. Blitzer et al. (2006) found it necessary to normalize and scale the new features obtained by applying the projection $\theta\mathbf{x}$. He argues that this was necessary to allow the new features "to receive more weight from a regularized discriminative learner". For each new feature, they centered the feature by subtracting off the mean and normalizing them to unit variance. Subsequently, they scaled the feature values by a factor α found on held-out data.

Restricted Regularization. When training the supervised model on the augmented feature space $\langle \mathbf{x}, \theta\mathbf{x} \rangle$, Blitzer et al. (2006) only regularize the weight vector of the original features, but not the one for the new low-dimensional features. They did this in order to encourage the model to use the new low-dimensional representation rather than the higher-dimensional original representation (Blitzer, 2008).

Dimensionality Reduction by Feature Type. An extension suggested in Ando and Zhang (2005) is to compute separate SVDs for blocks of the matrix W corresponding to certain feature types (as illustrated in Figure 5.6). Subsequently, separate projections are applied for every feature type submatrix. Due to the positive results in Ando (2006), Blitzer et al. (2006) include this in their standard setting of SCL and report results using block SVDs only.

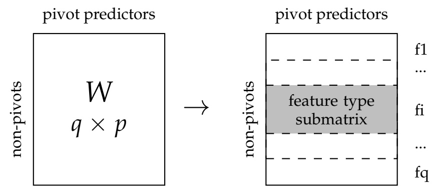


Figure 5.6: Illustration of dimensionality reduction by feature type. The gray area corresponds to a feature type (submatrix of W) on which the SVD is computed (block SVD), the white area is regarded as fixed to zero matrices.

5.3 Experimental Setup

This section describes the experimental setup including the construction of target domain data from Wikipedia.

5.3.1 Tools and Experimental Design

The baseline (source domain) disambiguation model is trained on the Alpino treebank (Cdb; newspaper text), which consists of approximately 7,000 sentences and 145,000 tokens (introduced in Chapter 2). For parameter estimation, in all experiments in this chapter we use the maximum entropy toolkit `tinyest` (introduced on page 24) with a Gaussian prior ($\mu = 0, \sigma^2 = 10,000$). We use `tinyest` also for training the binary pivot predictors. To compute the SVD, we use SVDLIBC.[4] The remaining parts of the algorithm are implemented in Python.

For self-training, it is necessary to define a way to pick the presumably 'correct' parse(s) out of the set of analyses of a sentence. This is done in order to get the positive and negative training instances for the parse disambiguation model. Therefore, for every sentence in the pool of unlabeled data, a subset of the parses of that sentence is marked as correct and the remaining parses are marked as incorrect. Since we do not have access to gold data to pick the presumably 'correct' parse, we have to rely on other mechanisms. In the experiments that follow, we use the out-of-domain model to score parses by calculating the probability of each parse under the source domain model. Then, those parses are marked as 'correct' that obtain the highest probability under that model. Obviously, the source model is not the best option, as it will suffer from domain shifts. However it turned out to be more stable than an alternative approach that randomly chooses one parse as the correct parse (out of all parses for a sentence). Furthermore, note that in all experiments in this chapter we restrict the maximum number of parses for the target domain data to be at most 200 (rather than 3,000, which is the standard setting to train the disambiguation component). This was done to limit the size of the target domain data for practical purposes. It did not impact the overall accuracy considerably, for instance, in the self-training setting (all-at-once) keeping all (up to 3,000) parses resulted in accuracy scores $\pm 0.5\%$ CA to those reported later (where only up to 200 parses were used).

For structural correspondence learning, we will report results of SCL with dimensionality parameter set to different values, $h = [25, 50, 100]$, and no feature-specific regularization or feature normalization/rescaling. We found that these additional steps did not improve results (Plank, 2009b). As target domain, we will consider the Dutch part of Wikipedia as data collection, described next.

5.3.2 Data: Wikipedia as Resource

Wikipedia is used both as test set and as unlabeled data source. Our assumption is that in order to parse data from a very specific domain, say about the artist Prince, then data related to that domain, like information about the new power generation, the purple rain movie, or other American singers and artists, should be of help. To gather this domain-specific target data, we exploit the category system of Wikipedia.

[4] Available at: `http://tedlab.mit.edu/~dr/svdlibc/`

Construction of Target Domain Data

We use the Dutch part of Wikipedia provided by WikiXML,[5] which is a dump of Wikipedia articles. As the corpus is encoded in XML, we can exploit general purpose XML Query Languages, such as XQuery, Xslt and XPath, to extract relevant information from WikiXML.

Given a Wikipedia page a and its associated categories, $c \in categories(a)$, we can identify pages related to a by various degrees of 'relatedness': directly related pages (those that share a category, i.e. all b where $\exists c' \in categories(b)$ such that $c = c'$). Additionally, we might include pages that share a sub- or supercategory of a, i.e. b where $c' \in categories(b)$ and $c' \in sub\_categories(a)$ or $c' \in super\_categories(a)$. For example, Figure 5.7 shows the categories extracted for the Wikipedia article about pope Johannes Paulus II.

```
<wikipage id="6729">
 <cat t="direct" n="Categorie:Paus"/>
 <cat t="direct" n="Categorie:Pools_theoloog"/>
 <cat t="super" n="Categorie:Religieus leider"/>
 <cat t="super" n="Categorie:Rooms-katholiek persoon"/>
 <cat t="super" n="Categorie:Vaticaanstad"/>
 <cat t="super" n="Categorie:Bisschop"/>
 <cat t="super" n="Categorie:Kerkgeschiedenis"/>
 <cat t="sub" n="Categorie:Tegenpaus"/>
 <cat t="super" n="Categorie:Pools persoon"/>
</wikipage>
```

Figure 5.7: Example of extracted Wikipedia categories for a given article (showing direct, super- and subcategories).

In order to create the set of related pages for an article a (that then constitutes the pool of unlabeled target data for a), we proceed as follows.

1. Find sub- and supercategories of the categories of article a
2. Extract all pages that are related to a (through sharing a direct, sub or super category)
3. Optionally, filter out certain pages

In our empirical setup, we follow Blitzer et al. (2006) and try to balance the size of source and target data. Thus, depending on the size of the resulting target domain data, and the "broadness" of the categories involved in creating

[5] Available at: `http://ilps.science.uva.nl/WikiXML/`

it, we might wish to filter out certain pages. We implemented a filter mechanism that excludes pages of a certain category (e.g. a supercategory that is hypothesized to be “too broad”). Alternatively, we might have used a filter mechanism that excludes certain pages directly. In the experiments, we always included pages that are directly related to a page of interest and those that shared a subcategory. Of course, the page itself is not included. With regard to supercategories, we usually included all pages sharing a supercategory with *a*, unless stated otherwise.

Test Data and Related Target Data

The test sets consist of a selection of biographical articles from Wikipedia. They were manually corrected in the course of the LASSY project (cf. page 27). An overview of the test sets including size indications is given in Table 5.1. The mean sentence length of the articles is slightly higher than the average newspaper sentence length of the source domain (which is 19.7 words).

Wiki/DCOI Id	**Article Title**	**Sentences**	**ASL**	**APS**
6677/26563	Prince (musician)	356	21.4	692.9
6729/36834	Paus Johannes Paulus II	237	22.3	780.9
182654/41235	Augustus De Morgan	254	24.1	1,004.3

Table 5.1: Size of test sets. ASL=average sentence length; APS=average parses.

Table 5.2 provides information on the data sets extracted from Wikipedia that constitute the pool of related target data for an article. For the Prince article, some supercategories were filtered out to obtain a data set of similar size to that of the source domain. Later, we will also use a larger data set that contains 357 articles, 13,838 sentences and 215,635 tokens.

Related to	**Articles**	**Sentences**	**Tokens**	**Relationship**
Prince	192	7,998	104,833	filtered super
Paus	418	9,370	176,208	all
De Morgan	274	6,797	113,984	all

Table 5.2: Size of related data gathered from Wikipedia. Relationship indicates whether all pages are used or some are filtered out (cf. Section 5.3.2).

5.4 Empirical Results

5.4.1 Baselines

To establish the baselines, we evaluate the standard Alpino model (trained on the out-of-domain source data) on the test sets.

Table 5.3 shows the baseline performance of the disambiguation model on the Wikipedia test articles. It provides concept accuracy (CA, i.e. the evaluation metric that was introduced in Section 4.3.2). The third and fourth column indicate the upper- and lower bounds for this task. They show that the baseline performance is already comparatively high (i.e. the baseline is much closer to the oracle than the random baseline that selects an arbitrary parse).

Article	**CA**	**Random**	**Oracle**
Prince (musician)	85.65	72.01	90.00
Paus Johannes Paulus II	86.59	73.64	91.16
Augustus De Morgan	83.30	69.45	86.91

Table 5.3: Baseline results: Performance of out-of-domain model trained on source domain (Cdb; newspaper text) on Wikipedia test articles. CA stands for concept accuracy (cf. Section 4.3.2). Random chooses an arbitrary parse per sentence. Oracle selects the best available parse per sentence (the parser is set to produce at most 3,000 analyses per sentence).

While the parser normally operates on an accuracy level of roughly 89%-90% on its own domain (newspaper text), the accuracy on the Wikipedia articles drops. The biggest performance loss (a drop to 83.30% CA) is observed on the article about the British logician and mathematician De Morgan. This confirms the intuition that this specific subdomain is the "hardest" of the three considered. For instance, the data contains mathematical expressions (e.g. *Wet der distributiviteit* 'distributivity law' *a(b+c) = ab+ac*). As shown in Table 5.1 (page 99), the ambiguity on the De Morgan test set is the highest with an average number of around 1,000 parses per sentence (APS), compared to around 700 for the other two test articles.

Overall, there are around 4-5% absolute parsing accuracy to be gained by adapting the parser to the specific target domains. The goal of the following section is to evaluate whether (and by how much) the performance of the system can be improved by exploiting unsupervised domain adaptation techniques to adapt the parse disambiguation component.

5.4.2 Self-training Results

First, we evaluate self-training with a single iteration and no selection. Afterwards, we will examine the setup with multiple iterations, where a specific part of the data is selected in each iteration.

Single Iteration, No Selection (all-at-once) The results for self-training with a single iteration (adding all automatically labeled data at once) are given in Table 5.4. All unlabeled data is labeled by the source domain model and added to the source data to train a new model. The out-of-domain source model is used to score the parses and the parse(s) with the highest probability is (are) marked as correct. The remaining parses are marked as incorrect and provide the negative training instances for the parse selection model.

Article	**CA**
Prince (all-at-once)	79.99 (-5.66)
Paus (all-at-once)	82.18 (-4.41)
De Morgan (all-at-once)	79.69 (-3.61)
Prince, more data (all-at-once)	78.89 (-5.76)

Table 5.4: Self-training results: single iteration, no selection. The performance of the self-trained model is considerably lower than the baseline (cf. Table 5.3, difference to baseline is given in parentheses).

The results show that simple self-training (that adds all automatically labeled data to the source domain data) does not help to improve the disambiguation component. The performance of the self-trained model is considerably worse than the baseline model trained on the labeled source data alone. Moreover, if we add more automatically labeled data (shown in the last row in Table 5.4 for the Prince data), accuracy drops even more.

The next question is whether other instantiations of self-training – i.e. ones that select instances from the pool of unlabeled data – are more effective. Therefore, we run several iterations of self-training where a certain amount of sentences are added in each iteration.

Multiple Iterations, Random Selection In this setup, in every iteration 500 sentences are randomly selected from the pool of unlabeled data and added to the labeled data to train a new model.

Multiple Iterations, Informed Selection In this setup, a scheme similar to that of Steedman et al. (2003) is followed. We first randomly select 1,500 sentences into a cache, and from the cache select 500 sentences. They are selected on the basis of either: (a) shorter sentence length; (b) fewer parses; or (c) lower entropy (where the source domain model is used for scoring).

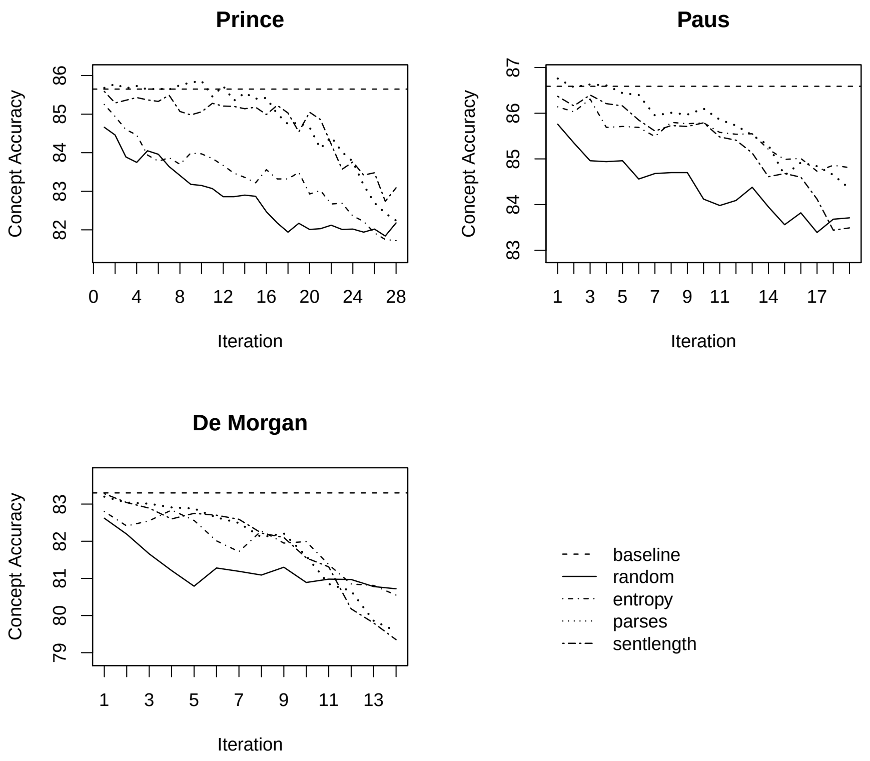


Figure 5.8: Results of self-training with multiple iterations. The straight dashed line is the source domain baseline. The figures show that self-training does not help. Only on one data set (Prince) a minor improvement over the baseline is first observed (by selecting sentences with fewer parses), however, performance then degrades. Random selection is worse than informed selection. No self-training variant yields positive results.

The results of evaluating multiple iterations of self-training with random and informed selection are shown in Figure 5.8. The straight dashed line in the figure represents the performance of the source domain model (baseline). The figures show that self-training does not help in this case either. A minor improvement over the baseline is first observed on only one data set (Prince; achieved by selecting sentences with fewer parses, i.e. less ambiguity, achieving its maximum of 86.86% CA in iteration 10, which is +0.21 absolute performance over the baseline). However, performance then degrades. Random selection is usually worse than informed selection. From the three informed selection mechanisms, entropy often performs worse than the other two.

We conclude that self-training does not work for adapting the disambiguation component of the Alpino parser. No self-training variant examined here improves the baseline. Instead, performance degrades when adding automatically labeled data. The obvious next question is whether the more involved technique that uses unlabeled data more implicitly is more effective.

5.4.3 Results with Structural Correspondence Learning

We tested our instantiation of SCL for parse disambiguation on the same three Wikipedia test sets as described above. Before looking at the results, we will examine the projection matrix to get a feeling about which feature correspondences were induced.

A look at θ To gain some insight into which kind of correspondences were learned, we examine the rows of θ. Recall that the inner product of a row i from matrix θ with a training instance gives a new real-valued feature. Let's denote it with $h_i = \theta_i \mathbf{x}$. This new feature is associated with a weight v_i that is trained from the augmented source data. If features have similar entries in the projection row θ_i (thus if their values in the row θ_i are similar), then they share a single weighting parameter v_i and thus are assumed to correspond. As shown in Blitzer (2008), this can be seen by expanding the inner product:

$$v_i h_i = v_i \theta_i \mathbf{x} = v_i \sum_j \theta_{ij} x_j \quad (5.3)$$

To illustrate the correspondences, let us look at two example rows from θ from the Prince data set. Figure 5.9 and 5.10 show excerpts from two projection rows. The first column represents the entry (value) of a feature, then follows the feature. The third column indicates whether it is a feature from the source domain (`src`), the target domain (`trg`) or a common feature (`both`).

Figure 5.9 shows an excerpt of features that obtained similar entries in row 21 of the projection matrix, which got the highest negative estimated weight. Feature correspondences under this projection row contribute towards dis-preferences for parses. As we can see, features presumably describing titles of songs got aligned with person names in the source data. However, the song titles were wrongly tagged as named entities describing persons.

```
0.00507587      f2('Valkenier',name('PER'))     src
0.00515761      f2('What Goes Around',name('PER'))      trg
0.00515761      f2('My Love',name('PER'))       trg
0.00515761      dep35('My_Love',name('PER'),hd/app,noun,single) trg
0.00515761      dep34('What_Goes_Around',name('PER'),hd/app,noun)       trg
0.00515761      dep34('My_Love',name('PER'),hd/app,noun)        trg
0.00513844      f2('Attje KeulenDeelstra',name('PER'))  src
0.00513844      dep34('Attje_KeulenDeelstra',name('PER'),hd/su,verb)    src
0.00525923      f2('Carlos Santana',name('PER'))        trg
0.00525923      dep35('Carlos_Santana',name('PER'),hd/obj1,prep,met)    trg
0.00528909      f2('Joop',name('PER'))  src
0.00528909      dep35('Joop',name('PER'),hd/su,verb,heb)        src
```

Figure 5.9: Example from θ (row 21) form the Prince data. The dimension got the highest negative estimated weight (-1.23).

```
0.00242024      appos_person('PER',jazz_zanger) trg
0.00242024      dep34('Jon_Hendricks',name('PER'),hd/app,noun)  trg
0.00242024      dep35(jazz_zanger,noun,hd/obj1,prep,met)        trg
0.00242024      depprep(verb,hd/mod,met,jazz_zanger)    trg
0.00242024      f2('Jon Hendricks',name('PER')) trg
0.00242024      f2('jazz-zanger',noun)  trg
0.00245022      appos_person('PER',bond_president)      src
0.00245022      f2(bondspresident,noun) src
```

Figure 5.10: Example from θ (row 16) from Prince data. This dimension of the projection matrix got the highest positive weight of +1.13).

In contrast, Figure 5.10 shows an extract of features from row 16 that got the highest positive estimated feature weight. Thus feature correspondences in this row are assumed to contribute to parse disambiguation. The algorithm aligned the apposition feature *jazz_zanger* 'jazz singer' to *bond_president* 'bond president' in the source domain. Thus, seeing 'jazz_zanger' at testing time would act as we had observed 'bond_president'. The question is whether such correspondences help for parse disambiguation, which we will examine next.

SCL Results Table 5.5 shows the results of SCL with varying h parameter (dimensionality parameter; $h = 25$ means that by applying the projection $\theta\mathbf{x}$, 25 new features are added to every source domain instance). Note that we use the same Gaussian regularization term ($\mu = 0$, σ^2=10,000) for all features (original and new features) and keep all entries in the pivot predictor matrix. As mentioned before, keeping only positive entries slightly degraded results. In an earlier study (Plank, 2009b) we also tested feature normalization and rescaling (as described in Section 5.2.2). While Blitzer et al. (2006) found it necessary to normalize (and scale) the projection features, we did not observe any improvement by normalizing them (actually, it slightly degraded performance in previous experiments). Therefore, results reported here are obtained without feature normalization/rescaling and with a single SVD (no block SVDs). However, we will look at block SVDs later.

	CA	
Prince baseline	85.65	
SCL, $h = 25$	85.75	(+0.10)
SCL, $h = 50$	85.73	(+0.08)
SCL, $h = 100$	85.69	(+0.04)
Paus baseline	86.59	
SCL, $h = 25$	86.72	(+0.13)
SCL, $h = 50$	86.67	(+0.08)
SCL, $h = 100$	86.51	(-0.08)
De Morgan baseline	83.30	
SCL, $h = 25$	83.07	(-0.23)
SCL, $h = 50$	82.96	(-0.34)
SCL, $h = 100$	82.95	(-0.35)

Table 5.5: Results of SCL (with varying h parameter, no feature normalization or rescaling, keeping all entries in matrix W). Difference to the baseline is given in parentheses.

The results in Table 5.5 show that structural correspondence learning only reaches a minor improvement in absolute parsing accuracy over the baseline in two out of the three test sets. On the De Morgan article, the performance is constantly below the baseline. By looking at individual accuracy scores per sentence, we noticed that parsing performance differs on a rather small subset of the test sentences. On the *paus* 'pope' test article, the results of SCL (with $h = 25$) and the baseline differ on 8 sentences only (out of 237 sentences). On five out of the 8, parsing accuracy was improved by SCL, while on 3 sentences it scored below the baseline. Similarly, on the Prince test set (which contains 356 sentences) scores differ on only 9 sentences, from which SCL did improve on 8 and scored below baseline on one sentence.

Overall, changing the dimensionality parameter h has a rather small impact on the results (cf. Table 5.5). This is in line with previous findings (Ando & Zhang, 2005; Blitzer et al., 2006). Therefore, the h parameter can be fixed to a small dimensionality (e.g. $h = 25$), which saves computing space and time.

To summarize, SCL reached a minor improvement in two out of the three test cases, but that was due to improved accuracy on a rather small set of sentences. The question is whether other instantiations of SCL are more effective. What happens if we use more data? And how does SCL perform with block SVDs (for feature subtypes)? These are the issues addressed next.

Note that the preliminary results reported in Plank (2009b) looked promising, despite the fact that the reported improvements were small. However, the extended evaluations presented in this chapter show that structural correspondence learning does not provide consistent and, more importantly, it does not provide significant improvements over the tough source domain baseline.

More Unlabeled Data In the experiments so far, we balanced the amount of source and target data. To examine the effect of more unlabeled target domain data, we extended the Prince data set by including additional related articles (increasing the unlabeled target data from 7,998 to 13,838 sentences). Table 5.6 shows the results of SCL with the larger data set. For $h = 25$ no difference can be observed, while for increasing h the performance of the models trained on the larger unlabeled data even (although only slightly) degrades. Thus, adding more unlabeled data did not give better results, either.

	CA
Prince baseline	85.65
SCL, $h = 25$	85.75
SCL, $h = 50$	85.73
SCL, $h = 100$	85.69
SCL with more target data, $h = 25$	85.75
SCL with more target data, $h = 50$	85.69
SCL with more target data, $h = 100$	85.55

Table 5.6: Result of SCL when more target data is used.

Dimensionality Reduction by Feature Type (Block SVDs) The last issue regarding SCL that we will examine is whether splitting up the predictor weight matrix by feature subtypes, and thus computing separate SVDs on blocks of the matrix, is more effective. In this way, there will be several θ's, one for each feature subtype. Moreover, a row in θ will encode correspondences between features from the same subtype only. Then, several projections will be computed, one for each feature subtype. This means that if there are k feature subtypes, the resulting training data will contain $k \times h$ new features.

We tested this extension of SCL by clustering nonpivots into five feature types: dependency relations (*dep*), PoS features (*pos*), bilexical relations (*z*), appositions (*app*) and syntactic features (*syn*, e.g. remaining grammar rules, coordination features etc.). For each feature type, a separate SVD was computed on the corresponding feature type submatrix (illustrated in Figure 5.6) and separate projections were applied to every training instance. Thus, our training instances now contain 125 new features (25 times 5). Moreover, we also report on results were a single submatrix is used at a time.

The results in Table 5.7 show that computing separate dimensionality reductions (block SVDs) for feature subtypes does not improve over the alternative (simpler) instantiation of SCL that uses the entire matrix W at once. Rather, the block SVD version of SCL performed worse, resulting in accuracies below the baseline. Thus, rather than helping, it degraded performance and it is therefore not worth performing this extra step.

Additionally, Table 5.7 shows the result of adding a single feature subtype at a time (e.g. only apposition features). From those, dependency or PoS features seem to help more than other feature subtypes. However, performance is still either below the baseline or reaches only a minor improvement.

Therefore, we conclude that structural correspondence learning does not work. It achieved only minor improvements that did not carry along all three test sets. In contrast, self-training was even worse. It used the unlabeled data in a much directer way. By adding the automatically labeled data to the seed data and retraining a model performance got worse. It scored considerably lower than SCL that uses the unlabeled data in a more indirect fashion.

	CA
Prince baseline	85.65
SCL, $h = 25$	85.75
SCL, block SVD, $h = 25$	85.66
SCL, only app, $h = 25$	85.66
SCL, only dep, $h = 25$	85.63
SCL, only pos, $h = 25$	85.73
SCL, only syn, $h = 25$	85.65
SCL, only z, $h = 25$	85.57
Paus baseline	86.59
SCL, $h = 25$	86.72
SCL, block SVD, $h = 25$	86.44
SCL, only app, $h = 25$	86.63
SCL, only dep, $h = 25$	86.70
SCL, only pos, $h = 25$	86.67
SCL, only syn, $h = 25$	86.44
SCL, only z, $h = 25$	86.56
De Morgan baseline	83.30
SCL, $h = 25$	83.07
SCL, block SVD, $h = 25$	83.22
SCL, only app, $h = 25$	79.47
SCL, only dep, $h = 25$	83.34
SCL, only pos, $h = 25$	83.07
SCL, only syn, $h = 25$	83.08
SCL, only z, $h = 25$	83.19

Table 5.7: Results of SCL with block SVDs (cf. Section 5.2.2) and adding a single feature subtype at a time. Computing separate projections for each feature subtype was not better than a single SVD.

5.5 Summary and Conclusions

This chapter presented an application of structural correspondence learning to parse disambiguation and compared it to the bootstrapping approach of self-training. While SCL has been successfully applied to PoS tagging and sentiment analysis (Blitzer et al., 2006, 2007), its effectiveness for parsing was rather unexplored.

Applying SCL involves many design choices and practical issues, which we tried to depict here in detail. Moreover, we examined several instantiations of self-training. However, the results are negative. The empirical findings show that self-training does not work for discriminative parse selection. None of the evaluated self-training variants (single versus multiple iterations, various selection techniques) worked: performance either improved only slightly, or degraded considerably most of the time. In case larger amounts of unlabeled data are added, the performance of the self-trained models dropped even more. Thus, only using the out-of-domain source model was more effective than self-training.

In contrast, the more indirect exploitation of unlabeled data through SCL seemed to be more fruitful than self-training, thus favoring the use of the more complex method. However, only a minor improvement on two out of three test cases could be achieved, and that was due to a few sentences only and not significant. Thus, SCL did not work either. Compared to self-training it did not degrade performance as much, but it is a much more complex technique, and so far, it is not really worth applying it to parse disambiguation.

Thus, adapting the discriminative parsing model using unlabeled data remains a hard task, and so far the best is to just use the available labeled data or simple supervised adaptation techniques.

Part III

Grammar-driven versus Data-driven Parsing Systems

Chapter 6

On Domain Sensitivity of Different Parsing Systems

In the last part (Part II, that covers Chapters 4 and 5), we exclusively focused on domain adaptation techniques for a syntactic disambiguation model, namely, the disambiguation component of the Alpino parser. In this chapter we extend the view to multiple parsing systems. The goal is to compare the performance of different types of parsing systems across domains. We do this in order to ask which parsing system is more affected by domain shifts, and thus more in need for domain adaptation techniques.[1]

6.1 Introduction

In the past decade, several natural language parsing systems have emerged which use different methods and formalisms. They fall broadly into two main categories: systems that employ a hand-crafted grammar with a statistical disambiguation component versus purely statistical data-driven systems. What they have in common is the lack of portability to new domains: their performance is likely to decrease substantially as the distance between test and training domain increases. Yet, to which degree do they suffer from this problem, i.e. which kind of parsing system is more affected by domain shifts?

Intuitively, a grammar-driven system should be less affected by domain changes. Rather, it might benefit from the combination of a well-established linguistic theory and a data-driven stochastic component (Zhang & Wang,

[1]This chapter is an extended version of Plank and van Noord (2010b), of which a part is also published in Plank and van Noord (2010a).

2009), where the hand-crafted grammar has usually been designed with broad coverage in mind. To investigate this hypothesis, an empirical investigation on Dutch is carried out. The performance variation of a grammar-driven parser, the Alpino parser (van Noord, 2006), versus two data-driven systems, the MST (McDonald et al., 2005) and Malt parser (Nivre et al., 2007), is evaluated across domains, and a simple measure to quantify domain sensitivity is proposed. This will give an estimate of which parsing system is more affected by domain shifts – and hence is more in need of adaptation techniques.

6.2 Related Work

For parsing, most previous work on domain adaptation has focused on data-driven systems (e.g. Gildea, 2001; McClosky et al., 2006; Dredze et al., 2007), i.e. systems that employ automatically-induced treebank grammars (cf. Chapter 2). Only few studies (e.g. Hara et al., 2005; Plank, 2009b) examined the issue of adapting the disambiguation component of a grammar-driven system. This may be motivated by the fact that potential gains for this task are inherently bound by the grammar. Yet, domain adaptation poses a challenge for both kinds of parsing systems. We therefore ask to what extent these different kinds of systems suffer from the problem. Intuitively, hand-crafted grammars should be less affected by domain shifts.

To the best of our knowledge, no study has yet addressed this issue of domain sensitivity of different parsing systems. Most previous work has focused on a single parsing system in isolation (e.g. Gildea, 2001; Hara et al., 2005; McClosky et al., 2006). However, there is an observable trend towards combining different parsing systems to exploit complementary strengths. For instance, Nivre and McDonald (2008) combine two data-driven systems (MST and Malt) to improve dependency parsing accuracy. Two other studies successfully combined grammar-based and data-driven systems: Sagae, Miyao and Tsujii (2007) incorporate dependencies obtained from a data-driven dependency parser as soft-constraints in a HPSG-based system to improve parsing accuracy on the Wall Street Journal. In the same spirit (but in the other direction), Zhang and Wang (2009) use a deep grammar-based backbone to improve data-driven dependency parsing accuracy. They incorporate features from the grammar-based backbone into the data-driven system to achieve better generalization across domains. As they evaluated their approach across several domains, we consider this work closest to ours.

In the following, we assess the performance variation of three dependency parsers on Dutch and propose a measure to quantify domain sensitivity.

6.3 Domain Sensitivity of Different Parsing Systems

The problem of domain dependence poses a challenge for both kinds of parsing systems, data-driven and grammar-driven. However, to what extent? Which kind of parsing system is more affected by domain shifts? We may rephrase the question as:

Q: Which parsing system is more robust with respect to different input texts?

To answer this question, we study the robustness of different parsing systems in terms of variation in accuracy on a variety of domains. The more the accuracy varies across domains, the less robust a given parsing system is. That is, robustness is an indicator for the sensitivity of a parsing system to different input texts.

6.3.1 Towards a Measure of Domain Sensitivity

We assume that we are given a parsing system p trained on some source domain (which we will call *baseline* hereafter), and given annotated data for a set of N target domains: $target = \{i|1 \leq i \leq N\}$. Each target domain is basically a data set or collection of texts from that specific domain. We can evaluate the performance (accuracy) of p on every $i \in N$ by comparing the parsed data to the gold standard data of the respective target domain. We thus assume that we have annotated data available for the N target domains in order to evaluate the performance of the different parsers. In the following, we will measure accuracy as LAS (labeled attachment score, introduced in Section 4.3.2).

The most intuitive measure of variation would be to simply calculate the standard deviation (*sd*) of the performance on the target domains.

LAS_p^i = accuracy of parser p on target domain i

$$sd_p^{target} = \sqrt{\frac{\sum_i (LAS_p^i - \mu_p^{target})^2}{|N| - 1}} \tag{6.1}$$

where:

$$\mu_p^{target} = \frac{\sum_i LAS_p^i}{|N|} \tag{6.2}$$

However, expressing variation in terms of standard deviation only will not provide us with information about the average parsing performance across domains. We will need both mean and standard deviation for measuring domain sensitivity. However, we have a concern here: plain mean and standard deviation do not take the source domain performance (baseline accuracy) into consideration nor the size of the target domain itself.

We propose to measure the domain sensitivity of a system by incorporating the source domain (baseline) performance, and weighting the various target domains by their size. Intuitively, we want a measure that indicates the average weighted gain or loss in performance relative to the source domain baseline. Moreover, the measure should give an indication of how much the performance of a parser fluctuates (its deviation around the baseline).

Let *average domain variation (adv)* be a measure consisting of two components: (i) the *net domain influence (ndi)*. It is the weighted average of the target performances with respect to the baseline, where w^i represents the size of target domain i and Δ_p^i is the difference in accuracy for that domain with respect to the baseline performance; and (ii) the *spread* of the performances with respect to the baseline (i.e. the standard deviation of the deltas).

More formally, *average domain variation (adv)* is defined as:

$$adv = ndi \pm spread \tag{6.3}$$

where:

$$ndi = \frac{\sum_i (w^i \times \Delta_p^i)}{\sum_i w^i} \tag{6.4}$$

with

$$\Delta_p^i = LAS_p^i - LAS_p^{baseline} \tag{6.5}$$

$$w^i = size(i) \tag{6.6}$$

and

$$spread = sd_p^{\Delta^{target}} \tag{6.7}$$

That is, we measure the *net domain influence (ndi)* as average performance relative to the baseline (source domain) by considering the non-squared differences from the out-of-domain (source domain) mean and weigh it by domain size. Note that in Equation 6.5 we do not take the absolute value of the performance difference on purpose. Keeping the sign was done in order to know whether the parser on average suffers a loss or gains with respect to the baseline. A consequence thereof is that the *ndi* measure might indeed be zero, even though the accuracy of two parsers varies a lot (as the differences might balance themselves out). In this case, the second part (*spread*) of the *average domain variation* measure still indicates which parser suffered more from domain shifts.

As an alternative to the *average domain variation* measure, we may want to just calculate a straight, unweighted average, the *unweighted average domain variation*: $uadv = \sum_i \Delta_p^i / |N| \pm spread$. However, this assumes that domains have a comparable and representative size, and a threshold might be needed to disregard domains that are presumably too small.

We will use *adv* in the empirical result section to evaluate the domain sensitivity of various parsing systems, where size will be measured in terms of number of words. We additionally provide values for the unweighted version using domains with at least 4,000 words (cf. Table 6.1).

6.4 Experimental Setup

6.4.1 Parsing Systems

The three parsing systems used in the evaluation are Alpino, a grammar-based parser for Dutch, and the two data-driven systems, MST and Malt. All three systems were introduced in Chapter 2. To recap, we use the non-projective algorithms of MST and Malt (Covington algorithm) to parse the Dutch data, and otherwise train the parsers using default settings. For example, we use the default first-order features for MST, and SVM with polynomial kernel and the default feature model for Malt. This means that the reported performance scores might be lower than state-of-the-art performance; on the other hand, the parsers are not tuned to a specific domain.

In the following, we will introduce the data sets, briefly discuss data conversion (from LASSY to CoNLL format) and give details on the employed evaluation scheme.

6.4.2 Source and Target Data, Data Conversion

The source domain on which all parsers are trained is Cdb, the newspaper part of the Eindhoven corpus (introduced in Section 2.4.1). For the cross-domain evaluation, we consider Wikipedia and the DPC (Dutch Parallel Corpus) as target data. All data sets are described next.

Source: Cdb To recap, the Cdb treebank consists of 140,000 words (7,136 sentences) from the Eindhoven corpus (newspaper text). It is a collection of text fragments from six Dutch newspapers. As described in Section 2.4.1, the collection has been annotated according to the guidelines of LASSY and is stored in XML format. Cdb is the standard treebank used to train the disambiguation component of the Alpino parser. Note that Cdb is a subset of the training corpus used for Dutch in the CoNLL 2006 shared task (Buchholz & Marsi, 2006). The CoNLL training data additionally contains a mix of non-newspaper text,[2] which we exclude here on purpose to keep a clean baseline.

Wikipedia	**Example article(s)**	**Articles**	**Words**	**ASL**
LOC (location)	Belgium, Antwerp (city)	31	25259	11.5
KUN (arts)	Tervuren school	11	17073	17.1
POL (politics)	Belgium elections 2003	16	15107	15.4
SPO (sports)	Kim Clijsters	9	9713	11.1
HIS (history)	History of Belgium	3	8396	17.9
BUS (business)	Belgium Labor Federation	9	4440	11.0
NOB (nobility)	Albert II	6	4179	15.1
COM (comics)	Suske and Wiske	3	4000	10.5
MUS (music)	Sandra Kim, Urbanus	3	1296	14.6
HOL (holidays)	Flemish Community Day	4	524	12.2
Total		**95**	**89987**	**13.4**

DPC	**Description/Example article(s)**	**Articles**	**Words**	**ASL**
Science	Medicine, oceanography	69	60787	19.2
Institutions	Political speeches	21	28646	16.1
Communication	ICT/Internet	29	26640	17.5
Welfare state	Pensions	22	20198	17.9
Culture	Darwinism	11	16237	20.5
Economy	Inflation	9	14722	18.5
Education	Education in Flanders	2	11980	16.3
Home affairs	Presentation (Brussel)	1	9340	17.3
Foreign affairs	European Union	7	9007	24.2
Environment	Threats/nature	6	8534	20.4
Finance	Banks (education banker)	6	6127	22.3
Leisure	Various (drugs-scandal)	2	2843	20.3
Consumption	Toys from China	1	1310	22.6
Total		**186**	**216371**	**18.5**

Table 6.1: Overview of the domains in the Wikipedia and DPC corpus, respectively (ASL = average sentence length).

[2]Namely, a large number of questions (from CLEF, roughly 4k sentences) and hand-crafted sentences used during the development of the Alpino grammar (1.5k).

Target: Wikipedia and DPC We use the Wikipedia and Dutch Parallel Corpus (DPC) subpart of the LASSY corpus[3] as target domains. These two corpora contain several subdomains, e.g. sports, locations, science. An overview of the corpora is given in Table 6.1. Note that both consist of hand-corrected data labeled by Alpino. This might introduce a slight bias towards Alpino, however, it has the advantage that all domains employ the same annotation scheme; note that different annotation schemes were the major source of error in the CoNLL 2007 shared task on domain adaptation so that no team was able to substantially improve performance over the in-domain baseline (Dredze et al., 2007).

CoNLL 2006 test file For a sanity check, we use the test data from the previous CoNLL shared task (CoNLL 2006 shared task on multi-lingual dependency parsing) to compare the data-driven models against state-of-the-art performance. This CoNLL 2006 test file consists of 386 sentences from an institutional brochure (about *jeugdgezondheidszorg* 'youth healthcare').

Data Conversion: LASSY to CoNLL format In order to train the MST and Malt parsers and evaluate them on the various Wikipedia and DPC articles, the first step is to convert the LASSY annotated files (which are in XML format) to the tabular CoNLL format (introduced in Chapter 2). To this end, we adapted the treebank conversion software developed by Erwin Marsi for the CoNLL-X 2006 shared task on multi-lingual dependency parsing. Instead of using the PoS tagger and tagset used in the shared task (which we initially did not have access to), we replaced the PoS tags with the tags obtained by parsing the data with Alpino.[4] At testing time, the data-driven parsers expect PoS tagged input, while Alpino expects plain sentences.

6.4.3 Evaluation

In all experiments, unless otherwise specified, performance is measured as labeled attachment score (LAS), the percentage of tokens with the correct dependency edge and label (cf. Section 4.3.2). To compute LAS, we use the CoNLL 2007 evaluation script[5] with punctuation tokens excluded from scor-

[3]LASSY (Large Scale Syntactic Annotation of written Dutch) project. Corpus version 17905, obtained from `http://www.let.rug.nl/vannoord/Lassy/corpus/`

[4]As will be discussed in Section 6.5 (cf. Table 6.2), using Alpino tags actually improves the performance of the data-driven parsers. We could perform this check as we later got access to the tagger and tagset used in CoNLL-X (Mbt with Wotan tagset; thanks to Erwin Marsi).

[5]Available at: `http://nextens.uvt.nl/depparse-wiki/SoftwarePage`

ing (the default setting in CoNLL 2006). We thus evaluate all parsers using the same evaluation metric. As discussed in Section 4.3.2, the standard metric for Alpino (concept accuracy) would be a variant of LAS, which allows for a discrepancy between expected and returned dependencies. Such a discrepancy can occur, for instance, because the syntactic annotation of Alpino/LASSY allows words to be dependent on more than a single head. However, such edges are ignored in the CoNLL format; just a single head per token is allowed. Furthermore, there is another simplification. As the Dutch tagger used in the CoNLL 2006 shared task did not have the concept of multiwords, the organizers decided to treat them as a single token (Buchholz & Marsi, 2006), e.g. `Vincent_van_Gogh` (multi-word-unit parts are joined with the underscore). In the following experiments, we follow the CoNLL 2006 task setup. To illustrate the main differences between the LASSY and CoNLL format, consider the following example:

(6.8) *Hierdoor werden drie Russische Ehboers gedood .*
Through this three Russian first-aiders were killed .

In LASSY, punctuation is attached to the head of the entire sentence. Thus, the sentence-final punctuation is attached to *werden*, resulting in the dependency triple ⟨werden,*punct*,.⟩. In CoNLL, punctuation is attached to the preceding non-punctuation token to reduce non-projectivity (as stated in a comment of the conversion tool written by Erwin Marsi). In the preceding example, this would mean that we instead get the following dependency relation: ⟨gedood,*punct*,.⟩. Moreover, the LASSY dependency parse contains an additional edge to represent that *Ehboers* 'first-aiders' is the direct object of *gedood* 'killed': ⟨gedood,*obj1*,Ehboers⟩. This additional edge is not present in the CoNLL format as it obeys the single-head property (Buchholz & Marsi, 2006). To evaluate the Alpino output, we first convert it to CoNLL format.

6.5 Empirical Results

6.5.1 Sanity Checks

First of all, we performed several sanity checks. We trained the MST parser on the entire original CoNLL training data as well as the Cdb subpart only, and evaluated it on the original CoNLL test data. As shown in Table 6.2 (row 1-2) the accuracies of both models fall slightly below state-of-the-art performance (row 5), most probably due to the fact that we used default parser settings (e.g. no second-order features for MST, default feature model for Malt). More importantly, there was basically no difference in performance when trained on the entire data or Cdb only.

Model	LAS	UAS
MST, trained on original CoNLL data	78.35	82.89
MST, trained on original CoNLL, Cdb subpart only	78.37	82.71
MST, trained on Cdb retagged with Alpino	82.14	85.51
Malt, trained on Cdb retagged with Alpino	80.64	82.66
MST, as reported in Nivre and McDonald (2008)	79.19	83.6
Malt, as reported in Nivre and McDonald (2008)	78.59	n/a
MST, trained on Cdb retagged with Mbt (used for CoNLL)	78.73	82.66
Malt, trained on Cdb retagged with Mbt (used for CoNLL)	75.34	78.29

Table 6.2: Performance of data-driven parsers versus state-of-the-art on the CoNLL 2006 test data (in labeled/unlabeled attachment score).

We then trained the MST and Malt parsers on the Cdb corpus converted into the retagged CoNLL format, and tested them on the CoNLL 2006 test data (also retagged with Alpino). As seen in Table 6.2, by using Alpino tags the performance level significantly improves ($p < 0.05$). This increase in performance might be attributed to two sources: (a) improvements in the Alpino treebank itself over the course of the years; (b) the PoS tags obtained by parsing the data with the Alpino grammar. To examine the contribution of each source, we trained an additional MST model on the Cdb data, but tagged with the same tagger as in the CoNLL shared task (Mbt, the Memory-based tagger developed by Daelemans, Zavrel, Berck and Gillis (1996), since we later got access to this tagger; cf. last row in Table 6.2). The results show that the major source of improvement actually comes from using the Alpino tags ($78.73 \rightarrow 82.14 = +3.41$ LAS), rather than the changes in the treebank ($78.37 \rightarrow 78.73 = +0.36$ LAS). Thus, despite the rather limited training data and use of standard training settings, we are in line with, and actually above, current results of data-driven parsing for Dutch.

6.5.2 Baselines

To establish our baselines, we perform 5-fold cross validation for each parser on the source domain (Cdb corpus, newspaper text). The baselines (in terms of labeled and unlabeled attachment score) for each parser are given in Table 6.3. Although absolute performance comparisons are not the goal here, the results show that the grammar-driven parser Alpino achieves a baseline that is significantly higher (90.75% LAS) in comparison to the data-driven systems (baselines around 80-83% LAS).

Model	**Alpino**	**MST**	**Malt**
Baseline (LAS)	90.76	83.63	79.95
Baseline (UAS)	92.47	88.12	83.31

Table 6.3: Baselines (5-fold cross-validation within Cdb, newspaper text). All differences between parsers are significant at $p < 0.05$.

6.5.3 Cross-domain Results

In order to assess the performance variation across domains, we evaluate each parser on the Wikipedia and DPC corpora that cover a variety of domains (described in Table 6.1).

Figure 6.1 and Figure 6.2 summarize the results for each corpus, respectively. In more detail, the figures depict for each parser the baseline performance as given in Table 6.3 (straight lines) and the performance on every domain (bars). Note that domains are ordered by size (number of words), so that the largest domains appear as bars on the left. If we instead measure performance in terms of unlabeled attachment score (UAS), we get very similar graphs, as shown in Figure 6.3. Therefore, in the following we report labeled attachment score (LAS) only.

Figure 6.1 depicts parser performance on the Wikipedia domains with respect to the source domain baseline. The figure indicates that the grammar-driven parser does not suffer much from domain shifts. Its performance is above the baseline for several Wikipedia domains. In contrast, the MST parser suffers the most from the domain changes; on most domains a substantial performance drop can be observed. The transition-based parser Malt scores overall considerably lower than MST and Alpino (in terms of absolute parsing accuracy), but seems to be less affected by the domain shifts.

We can summarize these findings by the proposed average domain variation measure (unweighted scores are given in the Figure): On average (over all Wikipedia domains), Alpino suffers the least ($adv = +0.81 \pm 2.9$), followed by Malt ($+0.59 \pm 2.9$) and MST (-2.2 ± 3.2), which on average loses 2.2 absolute LAS and shows the highest variation in performance. Thus, the graph-based data-driven dependency parser MST suffers the most.

We also evaluate the parsers on the more varied DPC corpus. It contains a broader set of domains, amongst others science texts (medical texts from the

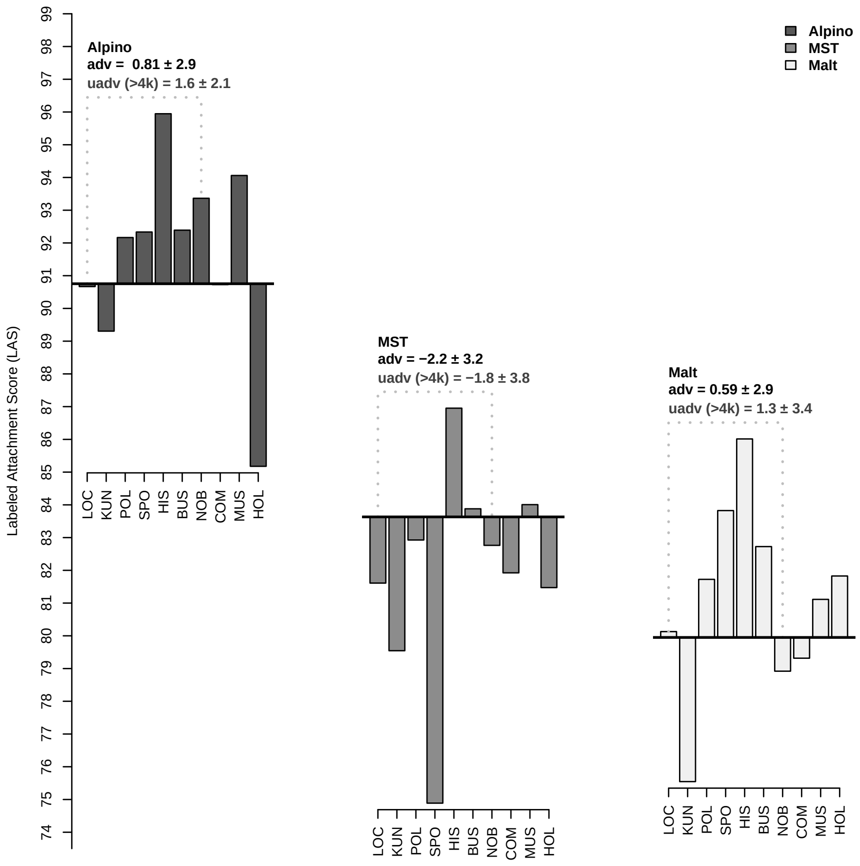


Figure 6.1: Performance on Wikipedia domains with respect to the source baseline (newspaper text, straight line) including average domain variation (adv) score and its unweighted alternative (uadv, calculated over domains with more than 4k words, indicated by the dotted rectangular). Domains are ordered by size (largest domains are on the left).

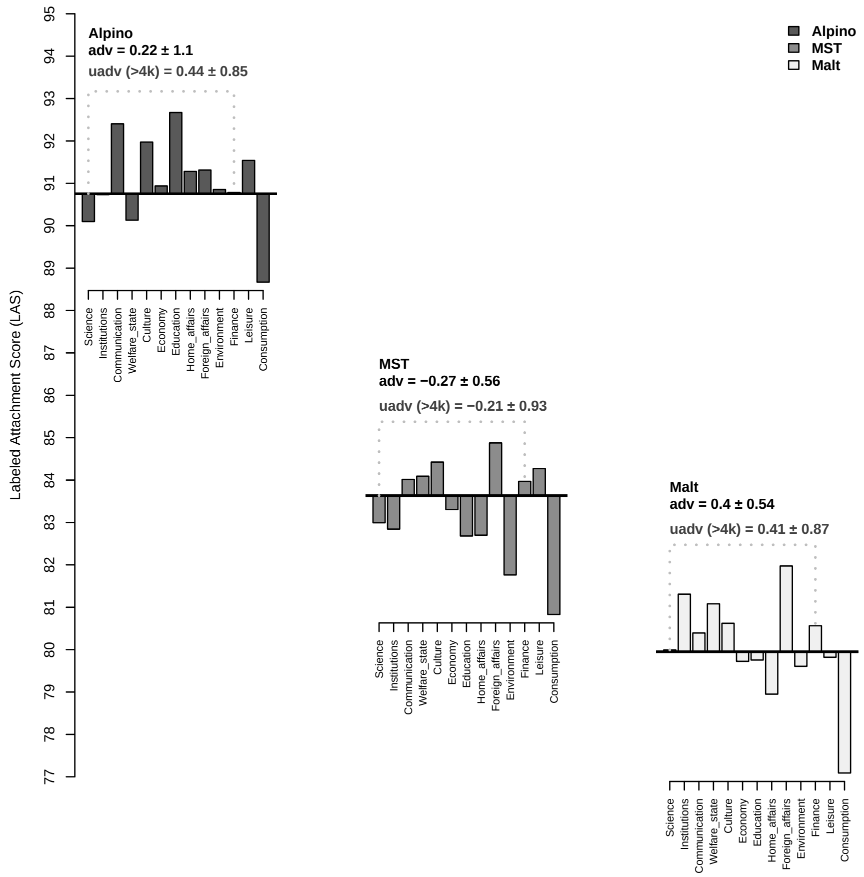


Figure 6.2: Performance on DPC domains with respect to the source baseline (newspaper text, straight line).

European Medicines Agency as well as texts about oceanography) and articles with more technical vocabulary (communication, i.e. internet/ICT texts). The results are depicted in Figure 6.2. Both Malt ($adv = +0.4 \pm 0.54$) and Alpino ($adv = +0.22 \pm 1.1$) achieve on average a gain over the baseline. As before, the graph-based data-driven parser MST is the most domain-sensitive parser also on DPC ($adv = -0.27 \pm 0.56$). In contrast, if we took only standard deviation of the parsing performance on the target domains into consideration (that disregards the baseline, as discussed in Section 6.3), we would get a completely opposite ranking on DPC: Malt would then be considered the most domain-sensitive parser (here higher sd means higher sensitivity): Malt ($sd = 1.20$), MST ($sd = 1.14$), Alpino ($sd = 1.05$). However, by looking at Figure 6.2, intuitively, MST suffers more from domain shifts than Malt. Most bars lie below the baseline. Moreover, the standard deviation measure neither gives a sense of whether the parser on average suffers a loss or gain over the new domains, nor does it incorporate the information of domain size. We thus conclude that the proposed average domain variation measure is a better suited measure.

To check whether the differences in performance variation are statistically significant, we performed an approximate randomization test (introduced in Section 4.3.2) over the performance differences (deltas) on the 23 domains (DPC and Wikipedia). The results show that the difference between Alpino and MST is significant. The same holds for the difference between MST and Malt. Thus, Alpino is significantly more robust than MST. However, the difference between Alpino and Malt is not significant. These findings hold for differences measured in both labeled and unlabeled attachments scores. Further, all differences in absolute performance across domains are significant.

To summarize, our empirical evaluation shows that the grammar-driven system Alpino is rather robust across domains. It is the best performing system and it is significantly more robust than MST. In contrast, the transition-based parser Malt scores the lowest across all domains, but its variation turned out not to be different from Alpino. Overall, MST is the most domain-sensitive parser.

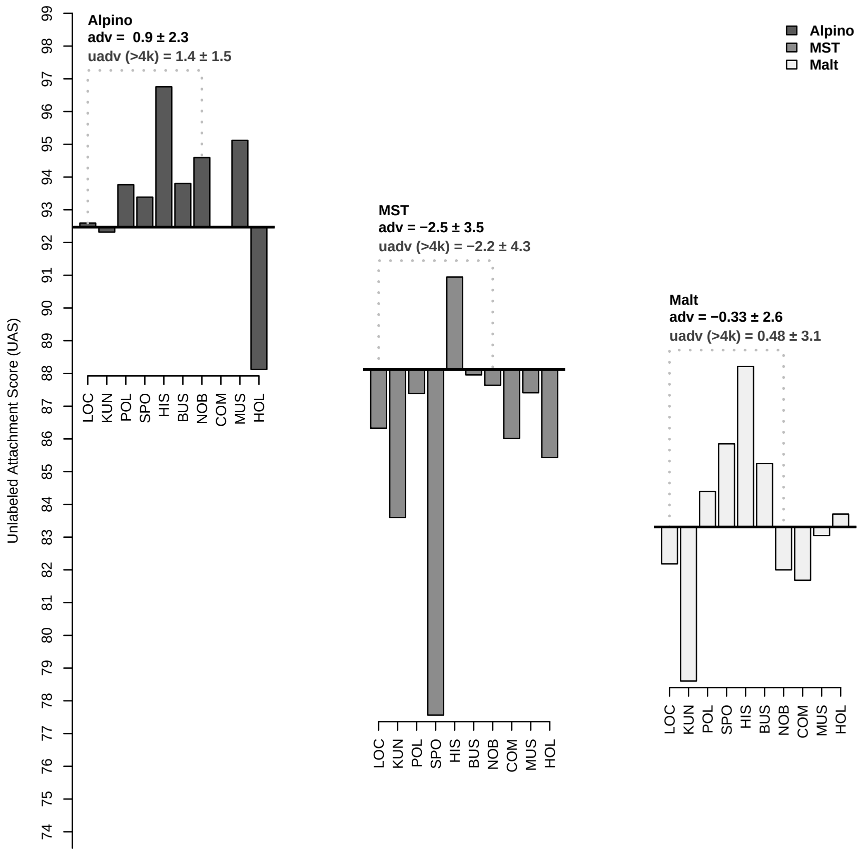


Figure 6.3: Performance on Wikipedia domains when accuracy is measured in unlabeled attachment score (UAS) – the resulting graph is similar to that of Figure 6.1 were labeled attachment score (LAS) is used, therefore in the remainder of this chapter we report LAS only.

6.5.4 Excursion: Lexical Information

Before discussing results of a qualitative error analysis, we will take a detour and evaluate the influence of lexical information. Both kinds of parsing systems rely on lexical information when learning their parsing (or parse disambiguation) model. However, to what degree do the parsers rely on lexical information, i.e. how much influence does lexical information have?

To examine this issue, we retrain all parsing systems by excluding lexical information. As all systems rely on a feature-based representation, we remove all feature templates that include words or stems and thus train models on a reduced feature space (original versus reduced space: Alpino 24k/7k features; MST 14M/1.9M features; Malt 17/13 templates). The unlexicalized models are evaluated on the same Wikipedia domains as before. The baseline is again 5-fold cross validation on the source domain (Cdb).

The result of evaluating the unlexicalized parsers on the Wikipedia domains is given in Figure 6.4, next to the performances of the lexicalized version to ease comparison. Obviously, absolute parsing performance drops for all parsers. In more details, lexicalized versus unlexicalized baseline performance on the source domain (in LAS) is, for each parser, respectively: Alpino $90.75 \rightarrow 89.36$, MST $83.63 \rightarrow 73.14$, Malt $79.95 \rightarrow 73.67$. As expected, performance drops to a higher degree for the data-driven parsers, but more for MST (-10.49 in LAS) than for Malt (-6.28 in LAS). In contrast, the variation in parsing performance across domains (average domain variation) remains similar. There is less variation for Alpino and MST, while more for Malt: Alpino $adv = +0.81 \pm 2.9 \rightarrow +0.77 \pm 2.1$, MST $adv = -2.2 \pm 3.2 \rightarrow -0.44 \pm 3.5$, and Malt $adv = +0.59 \pm 2.9 \rightarrow +1.3 \pm 3.1$. Interestingly, despite the fact that Malt was scoring lowest in absolute parsing performance in the lexicalized case, Malt is slightly better than MST (in absolute LAS) on all domains (including the source baseline) when running in this unlexicalized mode.

From the previous empirical results we know that the MST parser is the most domain-sensitive parser. The experiment presented in this section seems to suggest that this domain sensitivity comes from its high reliance on lexical information. When lexical information is omitted, the MST parser suffers the most: Its absolute performance level drops by -10.49% to 73.14 LAS, and thus even below the unlexicalized baseline of Malt of 73.67 LAS. The performance of the Malt parser dropped also, but less in comparison to MST. In contrast, Alpino suffers less from the missing lexical information.[6]

[6]Note that Alpino has still access to its lexicon here; we removed the lexicalized features from the trainable part of Alpino, the statistical disambiguation component.

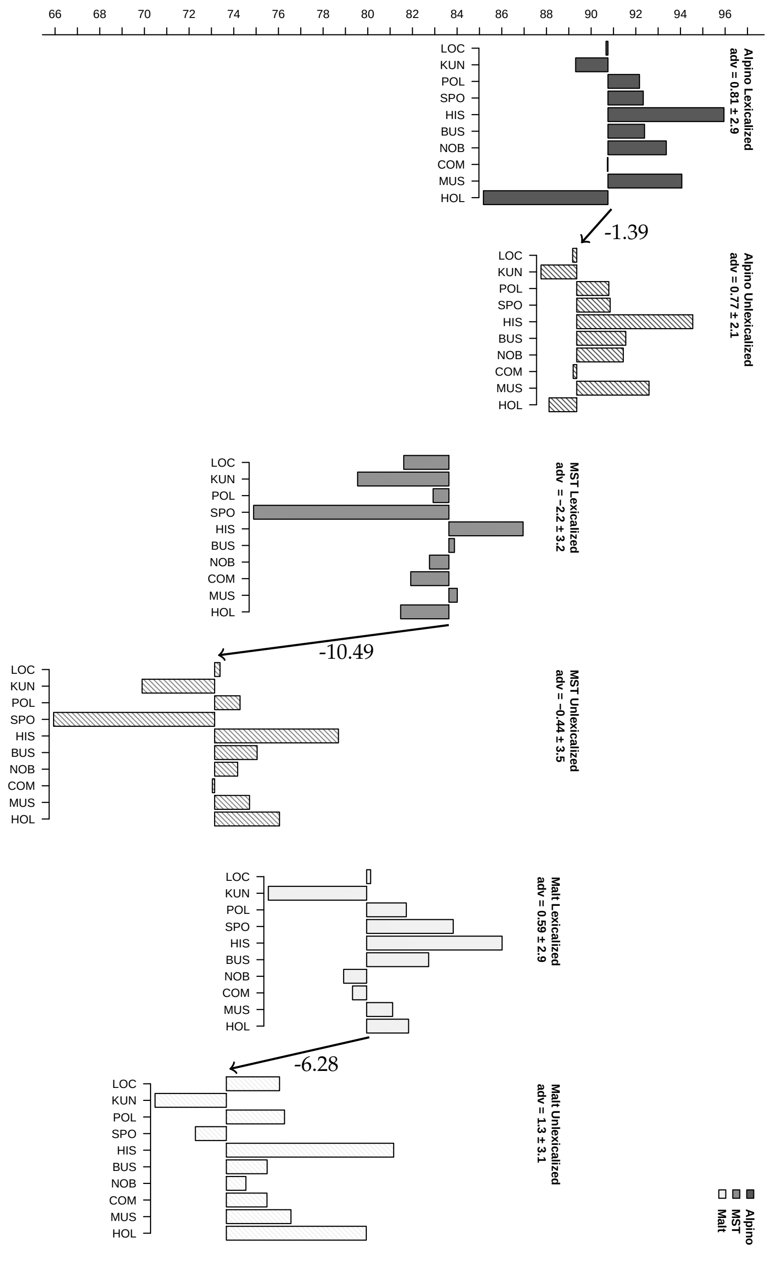


Figure 6.4: Performance of lexicalized versus unlexicalized parsers on Wikipedia domains.

6.6 Error Analysis

Before concluding this chapter, we present the results of a qualitative error analysis that relates parsing accuracy to several aspects of the input or predicted/gold dependency graphs. Such aspects include, as proposed in e.g. McDonald and Nivre (2007), factors of length like sentence length or dependency length, and linguistic factors like dependency type. We will compare the distribution of these aspects in a specific target domain to that of the original source domain (in the following abbreviated as SRC). The goal is to find out what might have caused the differences in parsing performance (presented in Section 6.5) between the parsers across domains.

Note that for the target domains we will mostly limit the analysis to the four largest domains per corpus (in terms of size) rather than all 23 subdomains, i.e. the leftmost bars in Figure 6.1 and 6.2 which include: location (LOC), arts (KUN), politics (POL) and sports (SPO) for Wikipedia; science, institutions, communication and welfare state for the DPC corpus.

6.6.1 Sentence Length

Sentence length is the number of tokens in a sentence. It is generally the case that parsing accuracy decreases with increasing sentence length, that is, longer sentences are usually harder to parse.

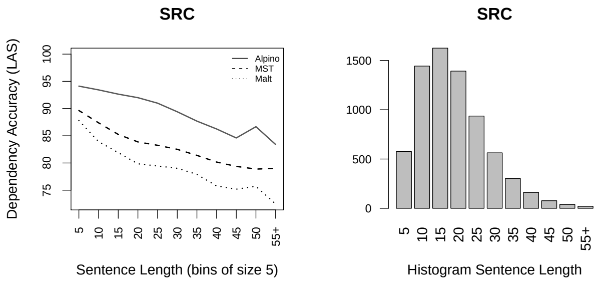


Figure 6.5: Accuracy relative to sentence length on the source domain (Cdb).

Figure 6.5 (left) shows the accuracy of the three parsers on the source (SRC) domain relative to sentence length in bins of size 5 (i.e. sentences of length 1-5, 6-10, etc.). Each bin contains at least 10 sentences (otherwise results are aggregated, indicated with +). The right part of the figure additionally shows the distribution over sentence lengths. Thus, the largest bin of SRC is the one that contains sentences between 11 and 15 words long. The general trend holds for all parsers: accuracy decreases with increasing sentence length. Similar figures result from the DPC subdomains and Wikipedia subdomains that we will not discuss here further. However, some Wikipedia subdomains exhibit a particularly different distribution, this will be discussed next.

On several Wikipedia domains, including location (LOC), arts (KUN), politics (POL) and sports (SPO), which are shown in Figure 6.6, the sentence length distribution is shifted to the left:[7] most sentences fall in the first bin, i.e. the majority of sentences are up to five words long. Interestingly, the three parsers exhibit quite different performance levels on this particular domains.

For instance, on the LOC (location) domain, shown in the upper-left part of Figure 6.6, we can see that the performance of the MST parser somewhat deviates from the performance of the other two parsers: it exhibits a low performance level on frequent short sentences, then increases, and decreases again. That is, MST has particular trouble in parsing these short sentences. In contrast, both Alpino and Malt follow the general trend (longer sentences are more difficult to parse, i.e. the performance almost steadily decreases). This is similar on the POL (politics) domain, but there Malt, too, scores low on the short sentences. On the sports (SPO) domain, the performance difference between MST and the other two parsers is particularly striking: here MST performs considerably worse on the short sentences compared to Alpino and Malt. In contrast, on the arts (KUN) domain, all parsers presumably have difficulties with parsing short sentences. In the following we will investigate possible causes of these differences.

By examining the data we found that Wikipedia domains often contain lists of various types. For instance, the LOC (location) articles contain lists of typical gastronomic specialties, or lists of majors (*burgermeesters*) or famous inhabitants. For example:

(6.9) *Mechels maantje : een chocolaatje*
'Mechels maantje': a chocolade [app]

(6.10) *Mechels torentje : een zandkoekje*
'Mechels torentje': a sand cake [app]

(6.11) *Marc_Didden , filmregisseur*
Marc_Didden, director [app]

[7]This is also the case for the BUS, NOB, COM Wikipedia subdomains not discussed here further. HIS, MUS and HOL follow the more standard distribution of sentence length similar to the SRC domain.

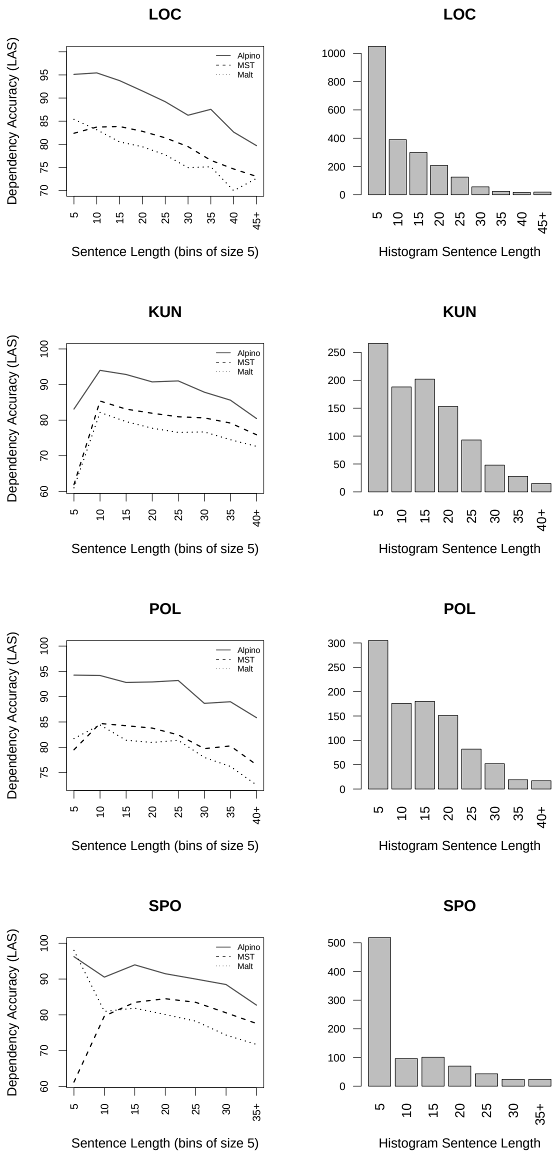


Figure 6.6: Accuracy relative to sentence length on Wikipedia domains.

(6.12) *Ludo_Van_Campenhout (VLD)* [mod]

In more detail, consider example 6.9, 6.10 and 6.11. MST considered *chocolaatje*, *zandkoekje* and *filmregisseur* as modifiers (mod) of *mannetje*, *torentje* and *Marc_Didden*, respectively, rather than apposition (app). Alpino correctly attached all of these appositions, while Malt produced fragmented trees (attached the descriptive noun phrase to the root token). Similarly, in example 6.12 Alpino correctly attached the party name (VLD) as modifier (mod) to the preceding name, while MST considered it a apposition and Malt attached it to the root token. Similar patterns can be found in the politics domain (POL), that contains list of politicians and parties with their abbreviations.

The sports domain (SPO) includes a particular long list (relative to the length of the articles) of winners of various races, e.g. winners of bike races:

(6.13) *2006 - Fabian_Cancellara*

(6.14) *2005 - Tom_Boonen*

(6.15) *2004 - Magnus_Bäckstedt*

(6.16) *2003 - Peter_Van_Petegem*

In the gold standard, both the name and the year are attached to the root token. This comes from the fact that in LASSY they are annotated as discourse parts (dp), while the CoNLL conversion software attaches discourse parts to the root token and labels them with the ROOT label. Alpino and Malt both correctly attached them to the root token, while MST wrongly considered the names as either modifiers (mod) or direct objects (obj1) of the year. Thus, MST gets them consistently wrong. As there are many of these fragments, this seems to be the main reason for the huge performance difference in the sports domain. On sentences up to 5 words, MST suffered a loss of -35% in absolute labeled attachment score compared to Malt and Alpino. In contrast, Malt often predicted the attachment correctly. This may be motivated by the fact that Malt has a tendency to overpredict root-modifiers "because all words that the parser fails to attach as modifiers are automatically connected to the root" (McDonald & Nivre, 2007). This, together with the artifact of the data concerning discourse parts, is presumably the reason that Malt performs considerably better in this domain than MST.

Regarding the arts (KUN) domain, the fact that all parsers score low on short sentences might be attributed to another peculiarity of the gold standard data. The arts domain contains several lists of names of artwork, such as the name of a painting (within quotes), followed by an optional year:

(6.17) *" De_boom ", 1959*

(6.18) *" Het balkon van Manet ", 1950*

(6.19) *" St-Jan_De_Doper "*

As we have already mentioned, Alpino attaches punctuation to the top node, while the CoNLL format attaches it to the preceding token. Thus, due to the conversion to CoNLL format, rather fragmented trees appear in the gold standard for the arts (KUN) Wikipedia domain. For instance, in the first example *"De_boom", 1959*, the opening quote as well as the multi-word unit *De_boom* are attached to the root. The rest (closing quote, comma and year) are all correctly attached to the multi-word unit, with 1959 being an apposition. Thus, the fact that all parsers score low on short sentences in this arts (KUN) subdomain can be attributed to artifacts of the data.

One might wonder what would happen to our domain variation analysis if we excluded these two particular subdomains, SPO and KUN. For completeness, Figure 6.7 shows the same graph as before (Section 6.5, Figure 6.1) but with SPO and KUN excluded. Still, MST is the most domain-sensitive parser, followed by Alpino and Malt, whose performance variations are similar on the remaining Wikipedia subdomains, despite the large difference in absolute parsing accuracy.

In summary, the sentence length distribution of the DPC and some of the Wikipedia subdomains resemble the distribution over sentence length of the source domain, which exhibits a more standard-shaped distribution (not skewed). In contrast, some particular Wikipedia domains deviate from it, and contain a particularly large portion of very short sentences. These short sentences are often lists of various sorts, and due to their high frequency, they often cause large performance differences between the parsers, either: a) because a parser consistently predicted the wrong dependency (like MST that predicted modifiers rather than appositions in LOC), or b) because of artifacts in the gold standard data (as is the case for SPO and KUN, e.g. the latter contains many fragmented trees due to the conversion that attached sentence-initial punctuation to the root token).

6.6.2 Dependency Length

An alternative length factor is dependency length (McDonald & Nivre, 2007). The dependency length of a dependency arc from word w_i to w_j is simply the absolute difference between the indices of the two words, i.e. $|i - j|$. Short dependencies typically represent modifiers of nouns, such as determiners or

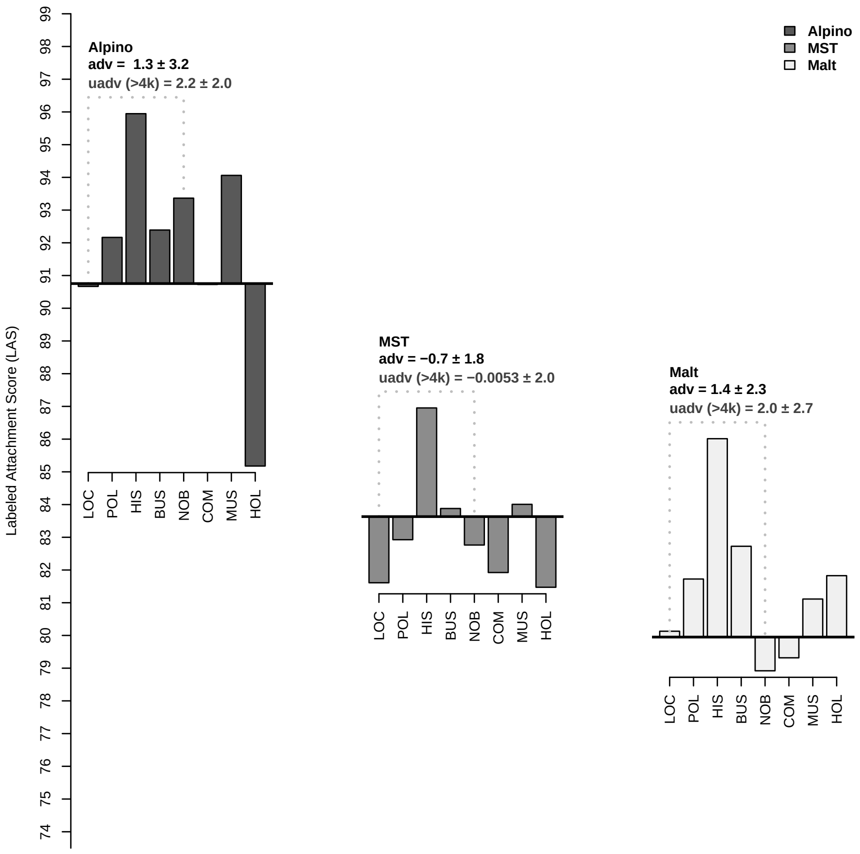


Figure 6.7: Performance on Wikipedia domains excluding SPO and KUN (cf. Figure 6.1 but without SPO and KUN subdomain).

adjectives (thus they are close to their dependent), while longer dependencies hold for example for modifiers of the main verb.

Figure 6.8 shows precision and recall relative to dependency length on the source domain (SRC). Precision is the percentage of predicted arcs of length *l* that were correct, while recall measures the percentage of gold standard relations of length *l* that were correctly predicted (McDonald & Nivre, 2007). Additionally, the rightmost graph gives the distribution of dependency length in the gold standard (the graphs for the individual parsers are very similar and thus not shown). Unsurprisingly, most dependency arcs are relatively short (i.e. are dependencies of length 1-2).

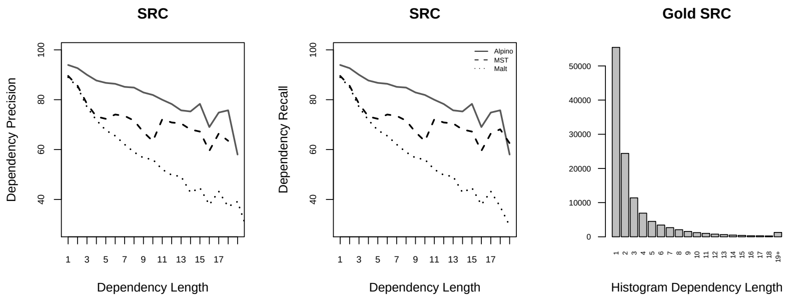


Figure 6.8: Dependency precision and recall. Right graph: Histogram of dependency length in the gold standard.

Dependency length did not provide any interesting insights in terms of differences between the source and the various target domains. The graphs for the individual Wikipedia and DPC subdomains looked similar to those in Figure 6.8. For completeness, we show the dependency precision and recall graphs for two subdomains in Figure 6.9. In general, the accuracy of MST and Malt is very similar on short dependencies. For longer dependencies, the MST parser (with its global search procedure) is usually more precise than Malt. This fact has already been noticed in McDonald and Nivre (2007), and we can confirm it here on the Dutch data.

6.6.3 Dependency Labels

Another interesting property is of a linguistic sort, namely, the type of the dependency label. In what follows, we analyze the dependency labels (also known as dependency types or dependency relations) in the source and the various target domains. An overview of the different dependency labels is given in Table 6.4.

Since we are using labeled attachment score, in the following a dependency edge is considered *wrong* if either the label, the head or both are incor-

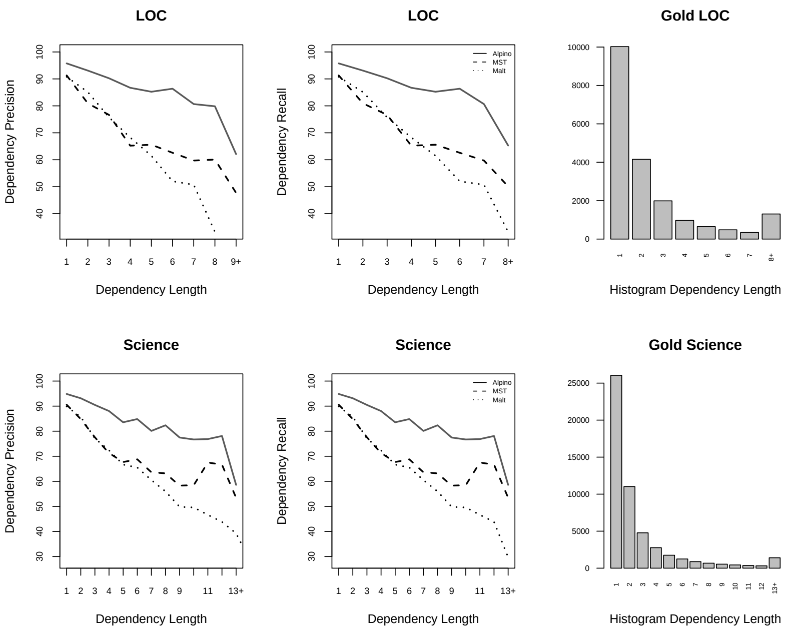


Figure 6.9: Dependency precision and recall relative to the gold standard.

rect. Alternatively, we might break down the error rate by head and labeling error, respectively. We decided not to do so, as it did not provide us with important additional insights.

In the following, we want to examine which dependency labels have most influence on parsing accuracy. In more detail, we will examine how much of the total error involves a certain dependency label, how frequent the label is in the gold standard and how difficult the label is. That is, we will provide three graphs per domain, such as those shown in Figure 6.10 for source (SRC), to illustrate:

(a) Overall error distribution by dependency label.

(b) Histogram of dependency labels in the gold standard.

Label	**Description**	**Example**
ROOT	head of the sentence	De stakers *zijn* wanhopig.
app	apposition	Jan_Tinbergen, *econoom*; de stad *Groningen*
body	body of complementizer	Hij zij dat *het bod is aanbevolen*.
cnj	conjunction	*oost* en *west*
crd	coordinator	noord *en* zuid; zowel lateraal *als* verticaal
det	determiner	*de* uitbreiding
hdf	final element of circumposition	want tot nu *toe*, van huis *uit*
ld	locative complement	het ligt *boven*
me	measure complement	het duurt *uren*, de *jaren* oude boerderij
mod	modifier	*veel* moeten schakelen
obcomp	comparison complement	de afdeling kan niet meer *dan* 16 opbrengen
obj1	direct object	Hilversum verwacht morgenavond weer een hoge *kijdichtheid*.
obj2	indirect object	hij vertelde *mij* dat
pc	prepositional object	vier miljard is bestemd *voor* de universiteiten
pobj1	preliminary direct object	Ik vind *het* goed dat...
predc	predicative complement	We zijn *uitwisselbaar*. De man in questie is een *Kongolees*.
predm	predicative modifier	Zij werd ernstig *gewond* in het ziekenhuis opgenomen.
punct	punctuation	Ja*!*
se	obligatory reflexive object	Zij voelden *zich* bedreigd
su	subject	*Zij* is nu thuis.
sup	preliminary subject	*het* was duidelijk, dat de liberalen ...
svp	particle of a verb	zakte hij *weg*, komt hij plotseling *binnen*
vc	verbal complement	Hij vraag zich af *of hij alles dan zo slecht heeft uigelegd*. Hij zij *dat het bod is aanbevolen*.

Table 6.4: Overview dependency labels in the Dutch CoNLL data.

(c) Error rate per dependency label.

Consider Figure 6.10. Graph (a) shows the overall error distribution per dependency label. That is, out of all errors, how often did it involve a certain dependency label. For example, 10% means that the dependency label accounted for 10% of the total error. This is obviously related to how frequent the label is. Therefore, graph (b) shows the distribution (histogram) of the dependency labels in the gold standard data. Moreover, to have an indication of how difficult a certain dependency label is, graph (c) shows the error rate per dependency label. That is, how often a certain dependency label was wrong or not predicted. This error rate is expressed in terms of inverted f-score of a label (100 - f-score). F-score is the harmonic mean of precision and recall of a dependency label. For instance, if a dependency label has an error rate of 60%, it means that it was wrong or not found in more than half of the cases.

For example, the predicative modifier (predm) relation is difficult. Figure 6.10 (c) shows that it was wrong in around 38% or 60% of the cases (for Alpino and the data-driven parsers, respectively). However, it is a rather infrequent label, as shown in Figure 6.10 (b), and hence its proportion from the total error is small. As can be observed in Figure (a), improving on that label will therefore not contribute much to overall parsing accuracy.

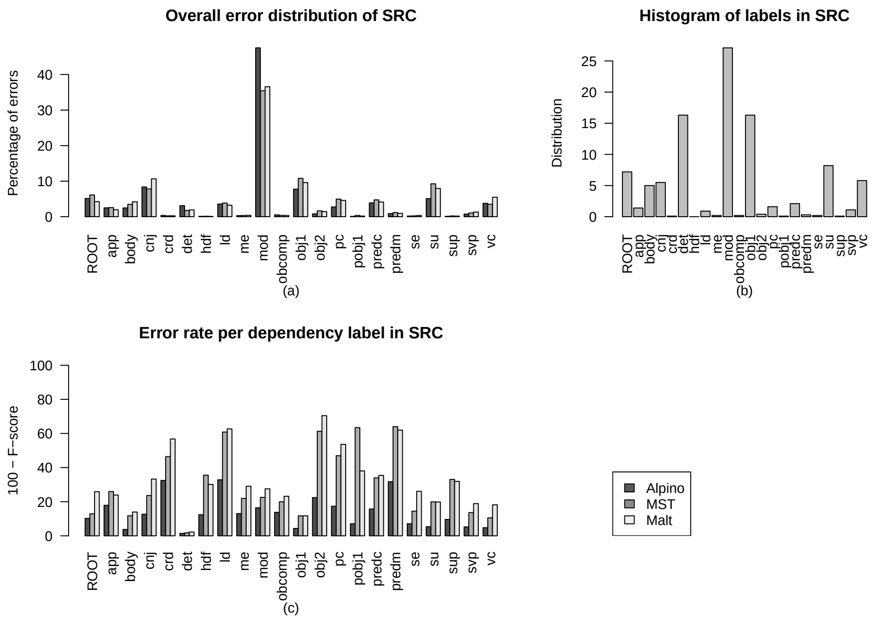


Figure 6.10: Dependency labels in SRC: (a) Overall error distribution; (b) the distribution of dependency labels in the gold standard; and (c) error rate per dependency label. For example, determiners (det) are frequent (b), however, they are not difficult (c) and therefore their proportion of the total error is small (a). In contrast, most of the errors involve modifiers (mod), as shown in graph (a). They are also the most frequent label (b) and relatively difficult (c).

Source domain Figure 6.10 provides information on dependency labels in the source domain (SRC). As can be seen from the overall error distribution, most errors in the SRC domain are modifiers (mod) errors. This is also the most frequent label, as shown in the right graph. However, compared to other dependency labels, modifiers are relatively difficult (in terms of error rate per label): the label was wrong in around 20-30% of the cases (in comparison, an even more difficult label is the locative complement with an error rate of 40-60%). The second most frequent labels are the direct object (obj1) and the determiner (det). However, determiners are easy compared to direct objects: as Figure 6.10 (c) shows, the label error rate for determiners is the overall lowest. In comparison, direct objects are wrong in approximately 5-10% of the cases.

Relations that were difficult but also relatively frequent in the SRC domain are: locative complements (ld), indirect objects (obj2), and conjunctions (cnj). All three parsers often confused locative complements with modifiers (such as *buiten* in Example 6.20 below). Indirect objects got confused with direct objects (e.g. in *Hij zal dit de regering voorschotelen* 'He will dish this up to the government' the direct object *dit* got confound with the indirect object *de regering*). Conjunctions (cnj) were mainly confused with modifiers or the root relation. For example, in the asyndetic conjunction *Ze zijn verouderd, kortzichtig* 'they are outdated, shortsighted', the latter token was considered a modifier (by Alpino and MST) and a predicative complement (by Malt) rather than a conjunct of *verouderd*. Finally, consider the sentence:

(6.20) *De heer Sleiffer werkt als bankemploye buiten Gorkum en de heer Breen is verbonden aan een suikerfabriek .*
'Mister Sleiffer works as banker outside Gorkum en mister Breen is affiliated with a sugar factory .'

Both MST and Malt wrongly considered the verb *werkt* as head of the entire sentence rather than the conjunction *en*, and thus got the conjunction wrong. Moreover, all three parsers considered *als bankemploye* as predicative modifier rather than modifier and, as already mentioned above, all of the parsers got the locative complement wrong (considered it as modifier of the verb *werkt*).

Next, we will analyze the dependency labels in the target domains. We will examine the overall error distribution and dependency label frequency in the target domain relative to the source domain (SRC), and relate it to the labeled error rate per dependency type in the target domain.

Wikipedia Figure 6.11 shows dependency label information for the location (LOC) domain in Wikipedia. To highlight interesting results, Figure 6.11 (a) shows only those dependency labels in LOC whose error distribution deviates more than 1% from the error distribution in SRC. The remaining graphs provide: (b) the dependency label distribution in LOC with respect to (w.r.t.) SRC; and (c) the labeled error rate per label in the LOC domain.

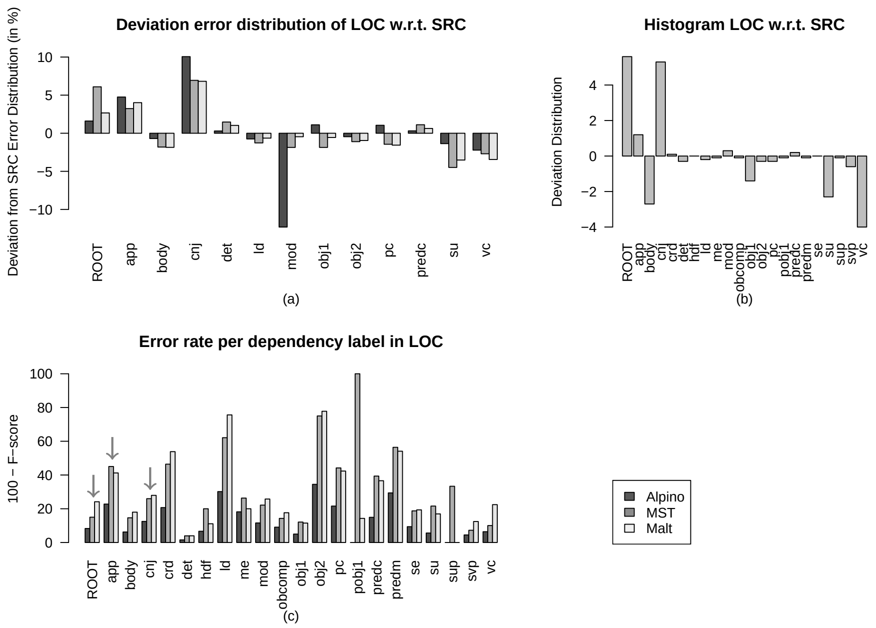


Figure 6.11: Dependency label information for the location (LOC) domain: (a) deviation of the error distribution of LOC with respect to SRC (for labels whose error rate deviates more that 1% from that of SRC); (b) the distribution of dependency labels in the gold standard of LOC with respect to SRC; and (c) error rate per dependency label. In the latter graph, those labels whose frequency is remarkably higher than in SRC (b) are marked with an arrow.

As shown in Figure 6.11 (b), the frequency of the ROOT, conjunction and apposition relation is much higher in the location (LOC) domain than in SRC, the domain on which the parsers were trained on. From those three labels (app, cnj, ROOT – highlighted with an arrow), Figure 6.11 (c) shows that the apposition relations was the most difficult one – its error rate was the highest (between 20 and 50%), despite the fact that appositions are less frequent than the root relation and conjunctions, cf. Figure 6.11 (c). This is a clear indicator that appositions are difficult in this Wikipedia domain. As we have already discussed in Section 6.6.1, the location articles often contain lists of gastronomic specialties, majors, or famous inhabitants with many appositions. Indeed, the MST parser often wrongly considered appositions as modifiers. This seems to be the reason for the considerable parsing performance difference between the parsers in this domain (the performance of MST dropped the most), as shown in Figure 6.1.

Also for the arts (KUN) and politics (POL) domains (not shown here as the graphs are similar), apposition, conjunctions and the ROOT relation account for the largest proportion of error, and again apposition was the most difficult from those three relations.

DPC While for the Wikipedia data appositions were particularly difficult (besides conjunctions and the ROOT relation), this was somewhat different on the DPC subdomains. In all four DPC domains considered here (science, institutions, communication and welfare state) apposition did not account for as large a proportion of the errors as in the Wikipedia case simply because appositions were less frequent in these subdomains (in fact, on all four DPC domains appositions were less frequent than in the source domain). Rather, most errors were made on conjunctions.

For instance, let us consider the science domain (which covers articles from oceanography and articles about medicines). Its dependency label information is shown in Figure 6.12 (again, somewhat similar graphs results from the other DPC subdomains not discussed here further). By inspecting the data, we found that the science articles contain many sentence fragments or complements consisting of complex noun phrases involving conjunctions. For instance, consider the following examples:

(6.21) *PAK- en metaalspecifieke biologische effecten*
'PAK- and metal-specific biological effects'

(6.22) *PCB's in biota en sedimenten*
'PCB's in biota and sediments'

(6.23) *Ariclaim is een blauwe (20 mg) of oranje (40 mg) capsule die de werksame stof duloxetine bevat.*
'Ariclaim is a blue (20 mg) or orange (40 mg) capsule that contains the working substance duloxetine.'

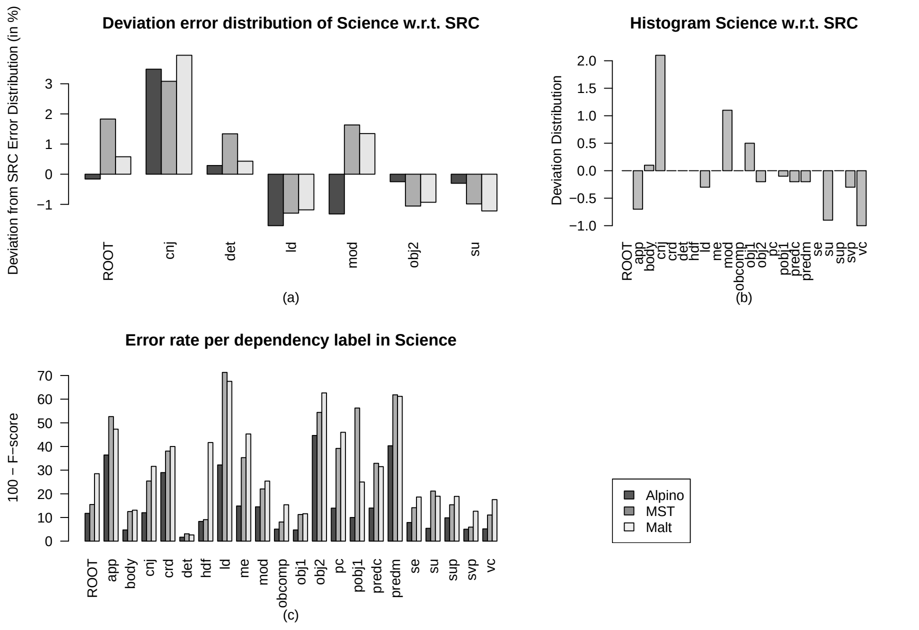


Figure 6.12: Dependency labels in the science domain: (a) deviation of the error distribution in science with respect to SRC (for labels whose error rate deviates more that 1% from that of SRC); (b) the distribution of dependency labels in the gold standard; and (c) error rate per dependency label.

In Example 6.21, *effecten* is the root of the fragment, which is modified by the conjunction *PAK- en metaalspecifieke* and the adjective *biologische*. Alpino correctly identified these relations, while Malt produced a fragmented tree: it attached both the conjunction and *effecten* to the root token, while correctly identifying the modifier *biologische*. We guess that this is the case because of the tendency of Malt to overproduce root attachments. Indeed, this can be seen from the high error rate on the ROOT relation of 30% (in fact, the Malt parser produced 5417 ROOT relations, while the gold standard contains only 3773, thereby obtaining low label precision). In contrast, MST considered the conjunction as head of the sentence, but considered both *metaalspecifieke* and *biologische* as modifiers of *effecten*, thus MST produced the most errors on this sentence. In Example 6.22, both MST and Malt correctly considered *in biota en sedimenten* as modifier of *PCB's*. In contrast, Alpino considered *PCB's* as coordinated with *sedimenten* and hence attached the conjunction *en* to the root token.

All three parsers had difficulty with identifying the correct head of the predicative complement in Example 6.23. Rather than considering *blauwe [...] of oranje [...]* as coordination that modifies *capsule*, Alpino and MST coordinated *blauwe [...] of [...] capsule* and considered the whole as predicative complement (besides additional errors with the parenthesized modifiers). Instead, Malt correctly coordinated *blauwe [...] of orange [...]*, but produced a fragmented tree and attached *is*, *of* and *capsule* to the root token.

To summarize, the modifier dependency relation is responsible for the most error in the SRC domain (newspaper text). It is the most frequent label, but it is also relatively difficult. Other frequent errors involved locative complements, indirect objects and conjunctions. In the Wikipedia domain, a particular difficult but also frequent relation was the apposition. For the DPC domains, all parsers had particular trouble with coordinations that were often part of complex noun phrases.

Of course, this error analysis could be further improved. For example, by adding an analysis of parsing accuracy relative to part-of-speech tag of the modifier, as done in McDonald and Nivre (2007). Alternatively, it would be interesting to inspect the biases of a parsing system in terms of structures (rather than single-edges) a parser tends to under- and overproduce, as recently proposed in Goldberg and Elhadad (2010).

6.7 Summary and Conclusions

In this chapter we examined a grammar-based system coupled with a statistical disambiguation component (Alpino) and two data-driven statistical parsing systems (MST and Malt) for dependency parsing of Dutch. By looking at the parser performance variation across a variety of domains, we addressed the question of how sensitive the parsing systems are to the text domain. We did this in order to gauge which kind of system is more affected by domain shifts, and thus more in need for adaptation techniques. We also proposed a simple measure to quantify domain sensitivity.

The results show that the grammar-based system Alpino is the best performing system, and it is robust across domains. In contrast, MST, the graph-based approach to data-driven parsing, is the most domain-sensitive parser. The results for the transition-based parser Malt indicate that its sensitivity is limited, but it is generally (in absolute terms) among the lowest scoring systems. In general, data-driven systems heavily rely on the training data to estimate their models. This becomes apparent when we exclude lexical information from the training process, which results in a substantial performance drop for the data-driven systems, MST and Malt, especially for MST. The grammar-driven model was more robust with respect to missing lexical information. Grammar-driven systems try to encode domain-independent linguistic knowledge, but usually suffer from coverage problems. The Alpino parser successfully implements a set of unknown word heuristics and a partial parsing strategy (in case no full parse can be found) to overcome this problem. This makes the system rather robust across domains, and, as shown in this study, significantly more robust than MST. This is not to say that domain dependence does not constitute a problem for grammar-driven parsers at all. As also noted by Zhang and Wang (2009), the disambiguation component and lexical coverage of grammar-based systems are still domain-dependent. Thus, domain dependence is a problem for both types of parsing systems, though, as shown in this study, to a lesser extent for the grammar-based system Alpino. Of course, these results are specific for Dutch; however, it is a first step. As the proposed methods are independent of language and parsing system, it would be interesting to apply them to other languages or systems.

Part IV

Effective Measures of Domain Similarity for Parsing

Chapter 7

Measures of Domain Similarity

As we have seen in the previous chapters, parsing accuracy degrades substantially when a model is applied to out-of-domain data (e.g. Table 3.1 on page 43). To overcome this problem, domain adaptation techniques attempt to exploit data that matches the domain of interest, as it is known (and we have discussed in Chapter 3) that in-domain data is the most beneficial data for training a parser for the new target domain (Sekine, 1997; Gildea, 2001). Hence, an important task is to select appropriate domains. However, most previous work on domain adaptation relied on the implicit assumption that domains are somehow given. That is, relevant (unlabeled or labeled) data was available and has usually been determined by a human. Knowing the target domain is of course a sensible scenario for specific application domains (e.g. constructing a parser for a dialogue system about train schedule inquiries). However, as more and more data becomes available, automatic ways to select data that is beneficial for a new (unknown) target domain are becoming attractive. For instance, consider parsing documents gathered from the web.

Therefore, the goal of this chapter is to evaluate ways to automatically acquire related training data for a given test set. The results show that an unsupervised technique based on topic models is effective – it outperforms random data selection on both languages examined, English and Dutch. It is closely followed by data selection using plain words only. Both automatic methods work better than selection based on manually assigned labels gathered from meta-data that is available for English. Moreover, in two out of three cases these automatic measures outperform a standard baseline model trained on the union of all data.[1]

[1]This chapter is an extended version of Plank and van Noord (2011).

7.1 Introduction and Motivation

Previous research on domain adaptation has focused on the task of adapting a system trained on one domain, say newspaper text, to a particular new domain, say biomedical data. For this new domain, usually some amount of (labeled or unlabeled) data was given – the amount and type of which has been determined by a human. The goal is to exploit the given data to adapt the system to the new domain (cf. Figure 7.1). This is also the setting that we assumed in Chapter 4 (where a limited amount of labeled data was available) and Chapter 5 (where only unlabeled data was given).

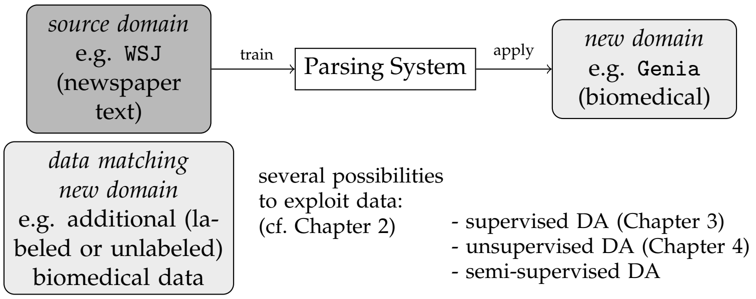

source domain
e.g. WSJ (newspaper text)
train
Parsing System
apply
new domain
e.g. Genia (biomedical)
data matching new domain
e.g. additional (labeled or unlabeled) biomedical data
several possibilities to exploit data: (cf. Chapter 2)
- supervised DA (Chapter 3)
- unsupervised DA (Chapter 4)
- semi-supervised DA

Figure 7.1: Example domain adaptation (DA) scenario. An implicit assumption of most prior work on DA is that domains are somehow ‘given’, i.e. there is some amount of (labeled/unlabeled) data available of the new domain.

However, with the growth of the web more and more data becomes available, where each document “is potentially its own domain” (McClosky et al., 2010). It is not straightforward to determine which data or model (in case we have several source domain models) will perform best on a new (unknown) target domain. The challenge is to decide how to best use the available sources to parse data from an unknown new domain. Therefore, an important issue that arises is *how to measure domain similarity*, i.e. whether we can find a simple and effective method to determine which model or data is most beneficial for an arbitrary piece of new text.

For parsing, to our knowledge only one recent study has started to examine this issue (McClosky et al., 2010). They coined the term *multiple source domain adaptation* therefore – we will discuss their approach in further detail in Section 7.2. Moreover, if we had such a measure of domain similarity, a related question is whether it can tell us something more about what is actually meant by domain. So far, it was mostly arbitrarily used to refer to some kind of coherent unit (related to topic, style or genre), e.g. newspaper text, biomedical abstracts, questions, fiction.

Most previous work on domain adaptation actually sidestepped this problem of automatic domain selection and adaptation. Rather, an implicit assumption of many previous studies on domain adaptation, e.g. Hara et al. (2005), McClosky et al. (2006), Blitzer et al. (2006), Daumé III (2007), is that domains are given, i.e. that they are represented by the respective corpora. Thus, a corpus has been considered a homogeneous unit. As more data becomes available, it is unlikely that domains will be 'given'. Moreover, a given corpus might not always be as homogeneous as originally thought (Webber, 2009; Lippincott et al., 2010). For instance, recent work has shown that the well-known Penn Treebank (PT) Wall Street Journal (WSJ) actually contains a variety of genres, including letters, wit and short verse (Webber, 2009).

In this study we take a different approach. Rather than viewing a given corpus as a monolithic entity, we break it down to the article-level and disregard corpus boundaries. Given the resulting set of documents (articles), we evaluate ways to automatically acquire related training data for a given test set, to find answers to the following questions:

- Given a pool of data (a collection of articles from unknown domains) and a test article, is there a way to automatically select training data that is relevant for the new domain? If so:
- Which similarity measure is good for parsing?
- How does it compare to human-annotated data?
- Is the measure also useful for other languages and/or tasks?

To this end, we evaluate measures of domain similarity and feature representations and their impact on dependency parsing accuracy. Given a collection of annotated articles and a new article that we want to parse, we want to select the most similar articles to train the best parser for that new article.

In the following, we will first compare automatic measures to human-annotated labels – that provide information on genre or topic of an article – by examining parsing performance within subdomains of the Penn Treebank WSJ. Then, we extend the experiments to the multiple domain adaptation scenario. Experiments were performed on two languages, English and Dutch. The empirical results show that a simple measure based on topic distributions is effective for both languages and works well also for part-of-speech tagging. It is closely followed by a technique that considers word distributions only.

As both approaches are based on plain surface-level information (words) and they find related data in a completely unsupervised fashion, they can be easily applied to other tasks or languages for which annotated (or automatically annotated) training data is available.

7.2 Related Work

The work most related to ours is McClosky et al. (2010). They try to find the best combination of source models to parse data from a new domain. The work closest to theirs is Plank and Sima'an (2008). In the latter, unlabeled data was used to create several parsers by weighting trees in the WSJ according to their similarity to the subdomain. The weights were based on probabilities given by language models trained on subdomain-specific unlabeled data. Given the resulting ensemble of subdomain sensitive parsers, they explored methods to amalgamate their predictions and showed that introducing domain sensitivity can improve over a tough, state-of-the-art baseline.

While Plank and Sima'an (2008) focus on the Penn Treebank WSJ only, McClosky et al. (2010) are the first to study the multiple source domain adaptation scenario for parsing. Given parsing models for each source domain, $m_1, \ldots, m_N$, their goal is to find the best mixing distribution, $\lambda_1, \ldots, \lambda_N$, of source domain models to parse data from an unknown target domain. Inspired by work on parsing accuracy prediction (Ravi, Knight & Soricut, 2008), they train a linear regression model to predict the best linear interpolation of source domain models m using the weights λ. In more detail, they train separate parsing models for each source domain. They use several measures of domain similarity/divergence (e.g. cosine similarity of the 50 most frequent words across domains, unknown word rate, the mixture weights themselves) as features in the regression model to predict the performance (parsing accuracy) of a given parsing model mixture on a new target domain. They then use the mixture with highest predicted performance to parse the new domain.

To create the training data for the regression model, they sample mixtures and evaluate these source domain mixture models on several target domains to obtain the input for the regression model (i.e. the f-scores of parsing model mixtures on different target data). Note that they computed the divergence scores over entire corpora (thus treated them as monolithic entities), rather than selecting or sampling subsets of data corresponding to the mixture weight. This implies that features would remain constant for different mixtures. Therefore, they divided the mixture weight by the score in order to integrate the mixture weight into the divergence/similarity score.

Thus, similar to us, McClosky et al. (2010) regard a target domain as mixture of source domains, but they focus on phrase-structure parsing. Furthermore, our approach differs from theirs in two respects: we do not treat source corpora as single entities in order to then mix models; instead, we consider articles as base units and try to find subsets of related articles (the most similar articles). Moreover, instead of creating a supervised model (in their case to predict parsing accuracy, which needs training data) our approach is 'simplistic': we apply measures of domain similarity directly (in an unsupervised fashion), without additionally training a supervised model.

Two other related studies are Lippincott et al. (2010) and Van Asch and Daelemans (2010). Van Asch and Daelemans (2010) explore a measure of domain difference (Renyi divergence) between pairs of domains and its correlation to PoS tagging accuracy. Their empirical results show a linear correlation between the measure and the performance loss. Their goal is different, but related: rather than finding related data for a new domain, they want to estimate the loss in accuracy of a PoS tagger when applied to a new domain. We will briefly discuss results obtained with the Renyi divergence in Section 7.5.1. Lippincott et al. (2010) examine subdomain variation in a biomedical corpus by quantitatively investigating a range of lexical and structural phenomena (e.g. sentence length, lexical differences). Their results show that there is rich variation between subdomains, thus suggesting the 'subdomain sensitivity' or awareness of NLP tools to subdomain variation, in a way related to Plank and Sima'an (2008). However, Lippincott et al. (2010) did not yet evaluate the effect of subdomain awareness on a practical task, thus our study is somewhat complementary to theirs.

The issue of data selection has recently been examined for language modeling (Moore & Lewis, 2010). A subset of the available data is automatically selected as training data for a language model based on a scoring mechanism that compares cross-entropy scores.[2] Their approach considerably outperformed random selection and two previous proposed approaches both based on perplexity scoring.

7.3 Measures of Domain Similarity

In the following, we will describe the feature representations and similarity functions that we considered for automatic training data selection. The section will end with the description of the meta-data that is available for the

[2]We tested data selection by perplexity scoring, but found the language models too small to be useful in our setting.

Penn Treebank WSJ. The meta-data provides information on genre or topic of articles in the WSJ, and thus allows for a comparison of the automatic data selection measures to selection using human-annotated labels.

7.3.1 Measuring Similarity Automatically

Feature Representations

A similarity function may be defined over any set of events that are considered to be relevant for the task at hand. For parsing, these might be words, characters, n-grams (of words or characters), part-of-speech (PoS) tags, bilexical dependencies, syntactic rules, etc. However, to obtain more abstract types such as PoS tags or dependency relations, one would first need to gather respective labels. The necessary tools for this are again trained on particular corpora, and will suffer from domain shifts, rendering labels noisy.

Therefore, we want to gauge the effect of the simplest representation possible: plain surface characteristics (unlabeled text). This has the advantage that we do not need to rely on additional supervised tools; moreover, it is interesting to know how far we can get with this level of information only.

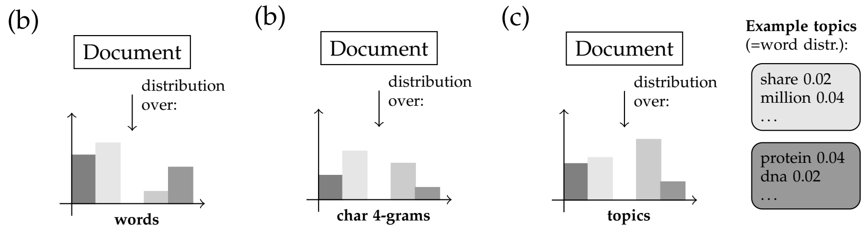


Figure 7.2: Feature representations for a document examined in this study: (a) probability distribution over words; (b) distribution over character tetragrams (4-grams); (c) distribution over topics, which in turn are probability distributions over words.

We examine the following feature representations, illustrated in Figure 7.2: relative frequencies of words, relative frequencies of character tetragrams, and topic models. Our motivation was as follows. Relative frequencies of words are a simple and effective representation used e.g. in text classification (Manning & Schütze, 1999), while character n-grams have proven successful in genre classification (Wu, Markert & Sharoff, 2010). Topic models (Blei, Ng & Jordan, 2003; Steyvers & Griffiths, 2007) can be considered an advanced model over word distributions: every article is represented by a topic distribution, which in turn is a distribution over words. Similarity between documents can be measured by comparing topic distributions.

Similarity Functions

There are many possible similarity (or distance) functions. They fall broadly into two categories: probabilistically-motivated and geometrically-motivated functions.

Probabilistically-motivated functions The well-known *Kullback-Leibler (KL) divergence* $D(q||r)$ is a classical measure of 'distance'[3] between two probability distributions, q and r, and is defined as:

$$D(q||r) = \sum_y q(y)(\log q(y) - \log r(y)) = \sum_y q(y) \log \frac{q(y)}{r(y)}$$

It is a non-negative, additive, asymmetric measure, and 0 iff the two distributions are identical. However, the KL-divergence is undefined (infinite) if there exists an event y such that $q(y) > 0$ but $r(y) = 0$, which is a property that "makes it unsuitable for distributions derived via maximum-likelihood estimates" (L. Lee, 2001).

One option to overcome this limitation is to apply smoothing techniques to gather non-zero estimates for all y. The alternative, examined in this study, is to consider an approximation to the KL divergence, such as the Jensen-Shannon (JS) divergence (Lin, 1991) or the skew divergence (L. Lee, 2001).

The *Jensen-Shannon (JS) divergence*, which is symmetric, computes the KL-divergence between q, r, and the average between the two functions. We use the JS divergence as defined in L. Lee (2001):

$$JS(q,r) = \frac{1}{2}[D(q||\text{avg}(q,r)) + D(r||\text{avg}(q,r))]$$

where $\text{avg}(q,r) = (q(y) + r(y))/2$.

The asymmetric *skew divergence* s_α, proposed by L. Lee (2001), mixes one distribution with the other by a degree defined by $\alpha \in [0,1)$ and is defined as:

$$s_\alpha(q,r,\alpha) = D(q||\alpha r + (1-\alpha)q)$$

As α approaches 1, the skew divergence approximates the KL-divergence.

[3]It is not a proper distance metric since it is asymmetric.

Geometrically-motivated functions An alternative way to measure similarity is to consider probability distributions as vectors and apply geometrically-motivated distance functions. This family of similarity functions include the *Euclidean*, *cosine* and *variational* distance functions.

The *variational* distance (also known as L1 norm, taxi-cab or Manhattan distance) is defined as:

$$var(q,r) = \sum_y |q(y) - r(y)|$$

It is zero if and only if $q(y) = r(y)$ for all y.

The *Euclidean* distance, also known as L2 norm, is the distance between the vectors defined as follows:

$$euc(q,r) = \sqrt{\sum_y (q(y) - r(y))^2}$$

Finally, *cosine* measures the angle between the two vectors. The cosine similarity is defined as:

$$cosine(q,r) = \frac{q(y) \cdot r(y)}{\|q(y)\| \|r(y)\|} = \frac{\sum_y q(y)r(y)}{\sqrt{\sum_y q(y)^2}\sqrt{\sum_y r(y)^2}}$$

Euclidean and variational are zero if and only if the two distributions are the same. In contrast, cosine achieves its maximum of 1 when $q(y) = r(y)$ for all y (L. Lee, 1997). For all other functions, it is the opposite: if the two distributions are the same, the function returns 0, and more than 0 otherwise.

An interesting relation holds between Jensen-Shannon, variational and Euclidean distance. As discussed in e.g. Lin (1991), L. Lee (1997) and Dagan, Lee and Pereira (1999), variational (L1) is an upper bound for the Jensen-Shannon divergence. Similarly, variational (L1) bounds the Euclidean (L2) distance, i.e. provides an upper bound for the Euclidean distance.

Example To give an illustration of the various similarity functions, consider a simple two-dimensional example taken from L. Lee (1997). Let the event set be $Y = \{y_1, y_2\}$. Moreover, for any probability distribution q, let $q(y_2) = 1 - q(y_1)$, and let r be a fixed uniform distribution: $r = (0.5, 0.5)$. Figure 7.3 provides a visualization of the various distance functions with respect to the fixed distribution r. The x-axis represents the probability of y_1 (thus, e.g. the point 0.8 on the x-axis represents the distribution $q = (0.8, 0.2)$). The figure shows that: i) skew divergence is indeed very close to Kullback-Leibler (the two lines almost overlap); ii) Kullback-Leibler is non-symmetric, as can be seen by the two plots of $KL(q||r)$ and $KL(r||q)$); iii) Jensen-Shannon is lower than (or equal to) variational for all values of $y1$; iv) Cosine reaches its maximum of 1 when the two distributions are the same, i.e. at $q = (0.5, 0.5)$, while all other functions are zero at that point. Note that if we would invert the cosine function (so that it also reaches its maximum at zero, i.e. adding -1), its line would end up between that of Jensen-Shannon and skew divergence.

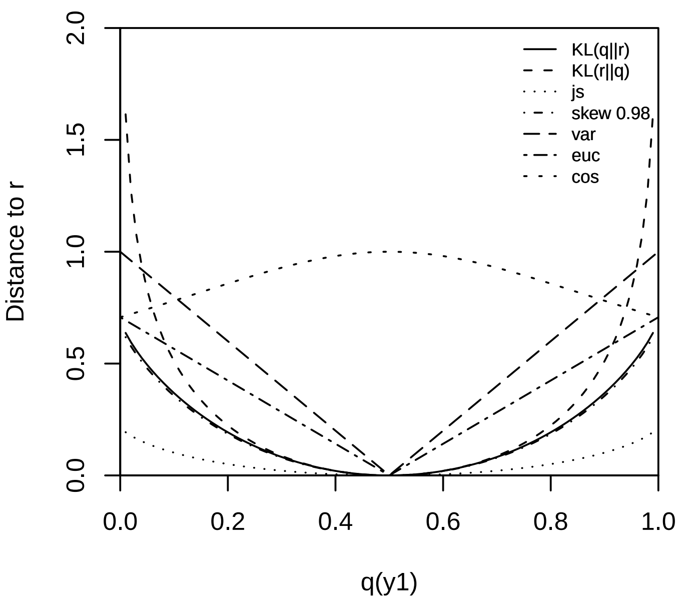


Figure 7.3: Comparison of similarity functions: distance from distribution $q = (y_1, y_2)$, with $q(y_2) = 1 - q(y_1)$, to fixed distribution $r = (0.5, 0.5)$. There is almost no difference visible between the Kullback-Leibler $KL(q||r)$ and skew divergence – indeed, the skew divergence approaches KL as α approaches 1.

7.3.2 Human-annotated Data

In contrast to the automatic measures devised in the previous section, we might have access to human-annotated data. That is, we might use label information such as topic or genre to define the set of similar articles. In general, we might see genre as reflecting more stylistic or syntactic differences, while topic will be more related to lexical differences between domains. These are surely two dimensions of variability that exist between domains. Therefore, it is interesting to examine the impact on parsing accuracy of the automatic measures and compare them to selection based on manually-annotated labels of topic and genre.

Genre For the Penn Treebank (PT) Wall Street Journal (WSJ), more specifically, the subset available in the Penn Discourse Treebank,[4] there exists a partition of the data by *genre* (Webber, 2009). Every article is assigned one of the following genre labels: news, letters, highlights, essays, errata, wit and short verse, quarterly progress reports, notable and quotable. The distribution over genre classes is given in Table 7.1. The largest portion of documents are news articles.

Errata	1% (25)
Essays	7.6% (166)
Highlights	2% (41)
Letters	3% (59)
News	85% (1855)
Notable & quotable	0.3% (6)
Quarterly progress reports	0.5% (12)
Wit and short verse	0.5% (15)

Table 7.1: Genre in the Penn Treebank WSJ subset of articles available in the Penn Discourse Treebank.

The genre classification has been made on the basis of meta-data (Webber, 2009). It is well-known that there is no meta-data directly associated with the individual WSJ files in the Penn Treebank. However, meta-data can be obtained by looking at the articles in the ACL/DCI corpus (LDC99T42), and a mapping file that aligns document numbers of the ACL/DCI corpus to WSJ keys (Webber, 2009). The genre classification has been made mainly on the basis of the headline information, e.g. the class 'Letters' comprises all and only

[4]`http://www.seas.upenn.edu/~pdtb/`

those files whose headline contains 'Letters to the Editor'.[5] An example document is given in Figure 7.4. The meta-data field HL contains headlines, SO source info, and the IN field includes topic markers. For example, the article in Figure 7.4 is a news article from the WSJ, whose headline is 'U.S. Savings Bond Sales. Suspended by Dept Limit'. The article was tagged with the following two topic markers: FINANCIAL, ACCOUNTING, LEASING (FIN) and BOND MARKET NEWS (BON).

```
<DOC><DOCNO>  891102-0186. </DOCNO>
<WSJKEY> wsj_0008 </WSJKEY>
<GENRE> news </GENRE>
<AN> 891102-0186. </AN>
<HL> U.S. Savings Bonds Sales
@  Suspended by Debt Limit </HL>
<DD> 11/02/89 </DD>
<SO> WALL STREET JOURNAL (J) </SO>
<IN> FINANCIAL, ACCOUNTING, LEASING (FIN)
BOND MARKET NEWS (BON) </IN>
<GV> TREASURY DEPARTMENT (TRE) </GV>
<DATELINE> WASHINGTON  </DATELINE>
<TXT>
<p><s>
The federal government suspended sales of U.S. savings bonds because
congress hasn't lifted the ceiling on government debt.
</s></p> [...]
```

Figure 7.4: Example of ACL/DCI article. We have augmented it with the WSJ filename (WSJKEY) and genre (GENRE) information.

Topic On the basis of the same meta-data, we devised a classification of the Penn Treebank WSJ by *topic*. That is, while the genre division has been mostly made on the basis of headlines, we use the information of the IN field. Every article is assigned one, more than one, or none of a predefined set of keywords. For example, the WSJ article in Figure 7.4 contains two topic markers, which we will also call IN fields. While the origin of these IN fields

[5]The classification of the individual files can be found at http://www.let.rug.nl/~bplank/metadata/genre_files_updated.html, see Section II., which is an updated version of Webber (2009) made in collaboration with Bonnie Webber. Many thanks!

remains unclear,[6] these keywords seem to come from a controlled vocabulary. In total, there are 76 distinct topic markers. The three most frequent keywords are: `TENDER OFFERS, MERGERS, ACQUISITIONS (TNM)`, `EARNINGS (ERN)`, `STOCK MARKET, OFFERINGS (STK)`. This reflects the fact that a lot of articles in the Penn Treebank WSJ come from the financial domain. But the corpus also contains articles from more distant domains, like `MARKETING, ADVERTISING (MKT)`, `PETROLEUM (PET)`, `HEALTH CARE PROVIDERS, MEDICINE, DENTISTRY (HEA)`.

Note that in the following, selection based on topics (IN fields) and topic model are two distinct approaches – the former is based on selecting articles based on overlaps of keywords given by the human-annotated labels (IN fields). The latter is based on comparing topic distributions of an automatically induced topic model that is estimated from the collection of documents using *latent dirichlet allocation* (LDA), described in the next section.

7.4 Experimental Setup

7.4.1 Tools and Evaluation

The parsing system used in this study is the MST parser (McDonald et al., 2005), a state-of-the-art data-driven graph-based dependency parser (introduced in Section 2.4.2). It is a system that can be trained on a variety of languages given training data in CoNLL format (Buchholz & Marsi, 2006). Additionally, the parser implements both projective and non-projective parsing algorithms. The projective algorithm is used for the experiments on English, while the non-projective variant is used for Dutch (as discussed in Chapter 2, sentences with a non-projective analysis are relatively infrequent in English in comparison to languages with fewer constraints on word order, like Dutch). Otherwise, we train the parser using default settings. MST takes PoS-tagged data as input; we use gold-standard tags in the experiments. We decided to take the MST parser for the experiments in this chapter as data-driven parsers are more in need for domain adaptation techniques (in comparison to the grammar-driven parser Alpino, cf. Chapter 6). Moreover, training MST (in non-projective mode) was faster than training Malt, and its overall accuracy level was higher. However, it would be interesting to test the approach also on other parsing systems. We leave this for future work.

[6]It is not known what IN stands for, as also stated in Mark Liberman's notes in the readme of the ACL/DCI corpus. However, a reviewer suggested that IN might stand for "index terms" which seems plausible.

We estimate topic models using latent dirichlet allocation (LDA, Blei et al., 2003) implemented in the MALLET[7] toolkit. The basic idea of topic models is that documents are mixtures of topics, where each topic is a distribution over words. A topic model is a generative model for documents (Blei et al., 2003; Steyvers & Griffiths, 2007). Each document in the corpus is a mixture of corpus-wide topics, and each word is drawn from one of those topics. Like Lippincott et al. (2010), we set the number of topics to 100, and otherwise use standard settings (no further optimization). We experimented with the removal of stopwords, but found no deteriorating effect while keeping them. Thus, all experiments are carried out on data where stopwords were not removed. However, we will later return and briefly discuss this issue of stopword removal.

We implemented the similarity measures presented in Section 7.3.1. For skew divergence, that requires parameter α, we set $\alpha = .99$ (close to KL divergence) since that has shown previously to work best (L. Lee, 2001). Additionally, we evaluate the approach on English PoS tagging using two different taggers: MXPOST (JMX), the maximum entropy tagger of Ratnaparkhi[8] and Citar,[9] a simple but fast trigram HMM tagger.

In all experiments, parsing performance is measured as labeled attachment score (LAS), the percentage of tokens with correct dependency edge and label (cf. Section 4.3.2). To compute LAS, we use the CoNLL 2007 evaluation script[10] with punctuation tokens excluded from scoring (as was the default setting in CoNLL 2006). PoS tagging accuracy is measured as the percentage of correctly labeled words out of all words. Statistical significance is determined by the approximate randomization test (introduced in Section 4.3.2) with 10,000 iterations.

7.4.2 Data

English - WSJ For English, we use the portion of the Penn Treebank Wall Street Journal (WSJ) that has been made available in the CoNLL 2008 shared task. This data has been automatically converted[11] into dependency structure, and contains three files: the training set (sections 02-21), development set (section 24) and test set (section 23).

[7]Available at: `http://mallet.cs.umass.edu/`

[8]Available at: `ftp://ftp.cis.upenn.edu/pub/adwait/jmx/`

[9]Citar has been implemented by Daniël de Kok and is available at: `https://github.com/danieldk/citar`

[10]Available at: `http://nextens.uvt.nl/depparse-wiki/`

[11]We use the LTH converter: `http://nlp.cs.lth.se/software/treebank_converter/`

Since we use articles as basic units, we actually split the data to get back original article boundaries.[12] This led to a total of 2,034 articles (1 million words). Further statistics on the data sets are given in Table 7.2. In the first set of experiments on WSJ subdomains, we consider articles from section 23 and 24 that contain at least 50 sentences as test sets (target domains). This amounted to 22 test articles.

	EN: WSJ	**WSJ+G+B**	**Dutch**
articles	2,034	3,776	51,454
sentences	43,012	77,422	1,663,032
words	1,049,211	1,784,543	20,953,850

Table 7.2: Overview of the data sets for English and Dutch. The Genia (G) and Brown (B) corpora were used in addition to the Wall Street Journal (WSJ) for the experiments of Section 7.5.2. The Dutch data is used in later experiments, and will be described in Section 7.6.1. However its size is given here for comparison.

To test whether we have a reasonable system, we performed a sanity check and trained the MST parser on the standard training sections (02-21). The result on the standard test set (section 23) is identical to previously reported results (excluding punctuation tokens: LAS 87.50, unlabeled attachment score (UAS) 90.75; with punctuation tokens: LAS 87.07, UAS 89.95). The latter is identical to what has been reported in Surdeanu and Manning (2010).

English - Genia (G) & Brown (B) For the domain adaptation experiments, we added 1,552 articles from the GENIA[13] treebank (biomedical abstracts from Medline) and 190 files from the Brown corpus to the pool of data. We converted the data to CoNLL format with the LTH converter (Johansson & Nugues, 2007). The size of the test files is, respectively: Genia 1,360 sentences with an average number of 26.20 words per sentence; the Brown test set is the same as used in the CoNLL 2008 shared task and contains 426 sentences with a mean of 16.80 words.

[12]This was a non-trivial task, as we actually noticed that some sentences from the original articles in the WSJ have been omitted in the CoNLL 2008 shared task data.

[13]We use the GENIA distribution of David McClosky that is available in Penn Treebank format at: `http://bllip.cs.brown.edu/download/genia1.0-division-rel1.tar.gz`

7.5 Experiments on English

7.5.1 Experiments within the Wall Street Journal

In the first set of experiments, we focus on the WSJ and evaluate the similarity functions to gather related data for a given test article. As discussed above, we have 22 WSJ test articles, sampled from sections 23 and 24. Regarding feature representations, we examined three possibilities: relative frequencies of words, relative frequencies of character tetragrams (both unsmoothed) and document topic distributions.

In the following, we mainly discuss representations based on words or topic models as we found character tetragrams less stable – We will briefly discuss them later, where we will see that character 4-grams performed sometimes like their word-based counterparts but other times, considerably worse.

Results of Automatic Similarity Measures

Table 7.3 compares the effect on parsing accuracy of the different ways to select training data in comparison to random selection. The table gives the average parsing accuracy over 22 test runs (one for each article, rather than showing individual tables for the 22 articles). We select training articles up to various thresholds that specify the total number of sentences selected in each round (e.g. 0.3k, 1.2k, etc.).[14] In more detail, Table 7.3 shows the result of applying various similarity functions defined over the two different feature representations (w: words; tm: topic model) for increasing amounts of training data. We additionally provide results of using the Renyi divergence.[15]

Clearly, as more and more data is selected, the differences become smaller, because we are close to the data limit for this set of experiments (all articles of the Penn Treebank WSJ). However, for all data points less than 38k (97%), selection by Jensen-Shannon, variational and cosine similarity outperform random data selection significantly for both types of feature representations (words and topic model). For selection by topic models, this additionally holds for the Euclidean measure.

From the various measures we can see that selection by Jensen-Shannon divergence and variational distance performs best, followed by cosine similarity, skew divergence, Euclidean and Renyi. The latter does not perform as well as other probabilistically-motivated functions. Regarding feature rep-

[14]Rather than choosing k articles, as article length may differ.

[15]The *Renyi divergence* (Rényi, 1961), also used by Van Asch and Daelemans (2010), is defined as $D_\alpha(q,r) = 1/(\alpha-1)\log(\sum q^\alpha r^{1-\alpha})$.

	1% (0.3k)	**2%** (0.6k)	**3%** (1.2k)	**6%** (2.4k)	**12%** (4.8k)	**25%** (9.6k)	**49%** (19.2k)	**97%** (38k)
random	70.61	74.00	77.21	79.52	81.64	82.98	84.48	85.51
w-js	74.07⋆	77.06⋆	79.41⋆	**81.50**⋆	82.69⋆	83.98⋆	84.94⋆	85.68
w-var	74.07⋆	**77.34**⋆	**79.60**⋆	81.13⋆	82.80⋆	83.82⋆	84.94⋆	85.45
w-skw	74.20⋆	76.65⋆	78.95⋆	80.72⋆	82.63⋆	83.68⋆	84.60	85.55
w-cos	73.77⋆	76.58⋆	79.30⋆	80.97⋆	82.68⋆	83.87⋆	84.96⋆	85.59
w-euc	73.85⋆	76.66⋆	78.90⋆	80.69⋆	82.34⋆	83.52⋆	84.68	85.57
w-ryi	73.41⋆	76.01⋆	78.31	80.78⋆	82.40⋆	83.76⋆	84.46	85.46
tm-js	74.23⋆	76.80⋆	79.49⋆	81.43⋆	82.86⋆	84.04⋆	85.01⋆	85.45
tm-var	**74.29**⋆	76.94⋆	79.59⋆	81.32⋆	**83.06**⋆	83.93⋆	84.94⋆	85.43
tm-skw	74.13⋆	77.10⋆	79.42⋆	81.49⋆	82.84⋆	84.13⋆	84.82	**85.73**
tm-cos	74.04⋆	76.70⋆	79.27⋆	81.19⋆	82.91⋆	**84.14**⋆	84.99⋆	85.42
tm-euc	74.27⋆	77.04⋆	79.53⋆	81.09⋆	82.88⋆	83.93⋆	**85.15**⋆	85.62
tm-ryi	71.26	75.85	78.64⋆	81.04⋆	82.50⋆	83.79⋆	84.85	85.58

Table 7.3: Parsing accuracy for increasing amounts of training data as average over 22 WSJ articles: Comparison of similarity measures based on words (w) and topic model (tm), where js=Jensen-Shannon, cos=cosine, skw=skew, var=variational, euc=Euclidean, and ryi=Renyi. Best per-group (feature representation) score is underlined; best overall score is in bold; ⋆ indicates that the result is significantly better ($p < 0.05$) than the random baseline.

resentations, the representation based on topic models works slightly better than the respective word-based measure (cf. Table 7.3) and often achieves the overall best score (cf. boldface).

Overall, the differences in accuracy between the various similarity measures are small; but interestingly, the overlap between them is not that large. Table 7.4 and Table 7.5 shows the overlap in terms of proportion of identically selected articles between pairs of similarity measures. As shown in Table 7.4, for all measures there is only a small overlap with the random baseline (around 10%-14%). Despite similar performance, topic model selection has interestingly no substantial overlap with any other word-based similarity measures: their average overlap is at most 41.6%.

Moreover, Table 7.5 compares the overlap of the various similarity functions within a certain feature representation (here x stands for either topic

	ran	w-js	w-var	w-skw	w-cos	w-euc
ran	–	10.3	10.4	10.0	10.4	10.2
tm-js	12.1	41.6	39.6	36.0	29.3	28.6
tm-var	12.3	40.8	39.3	34.9	29.3	28.5
tm-skw	11.8	40.9	39.7	36.8	30.0	30.1
tm-cos	14.0	31.7	30.7	27.3	24.1	23.2
tm-euc	14.6	27.5	27.2	23.4	22.6	22.1

Table 7.4: Average overlap (proportion of identically selected articles in %) of similarity measures: random selection (ran) vs. measures based on words (w) and topic model (tm).

	tm/w-js	tm/w-var	tm/w-skw	tm/w-cos	tm/w-euc
tm/w-var	76/74	–	60/63	55/48	49/47
tm/w-skw	69/72	60/63	–	48/41	42/42
tm/w-cos	57/42	55/48	48/41	–	62/71
tm/w-euc	47/41	49/47	42/42	62/71	–

Table 7.5: Average overlap (proportion of identically selected articles in %) of similarity measures for different feature representations x as tm/w, where tm=topic model and w=words. Highest pair-wise overlap is underlined.

model – left value – or words – right value). The table shows that there is quite some overlap between Jensen-Shannon, variational and skew divergence on one side, and cosine and Euclidean on the other side, i.e. between probabilistically- and geometrically-motivated functions. Variational has a higher overlap with the probabilistic functions. Interestingly, the 'peaks' in Table 7.5 (underlined, i.e. the highest pair-wise overlaps) are the same for the different feature representation. For example, the Euclidean distance function shares the most with cosine for both topic model (62% overlap) and the word-based representation (71% overlap).

In the following, we will consider selection by topic model and words, as they are relatively different from each other, despite similar performance. For the word-based model we use Jensen-Shannon as similarity function, as it turned out to be the best measure. For topic model, we use the simpler variational metric. However, very similar results were achieved using Jensen-

Shannon. Cosine and Euclidean did not perform as well. In later section, we will also examine a simple technique for combining these two measures.

Automatic Measure vs. Human labels

The next question is how these automatic measures compare to data selection based on human-annotated data. We compare word-based and topic model-based selection (by using Jensen-Shannon and variational, respectively) to selection based on human-given labels: genre and topic. For genre, we randomly select larger amounts of training data for a given test article from the same genre. For topic, the approach is similar, but as an article might have several topic markers (keywords in the IN field), we rank articles by proportion of overlapping topic keywords.

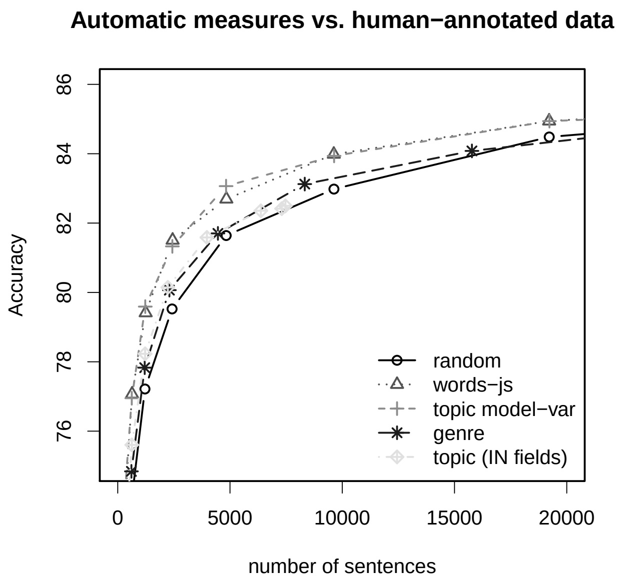


Figure 7.5: Comparison of automatic measures (words using Jensen-Shannon and topic model using variational) with human-annotated labels (genre/topic). Automatic measures outperform human labels ($p < 0.05$).

Figure 7.5 shows that human-labels do actually not perform better than the automatic measures. Both are close to random selection. Moreover, in the graph the curve of selection by topic marker (IN fields) stops early – we believe the reason for this is that the IN fields are too fine-grained, which limits the number of articles that are considered relevant for a given test article. However, manually aggregating articles on similar topic markers did not improve topic-based selection either. We conclude that the automatic selection techniques perform significantly better than human-annotated data, at least within the WSJ domain considered here.

7.5.2 Domain Adaptation Results

Until now, we compared similarity measures by restricting ourselves to articles from the WSJ. In this section, we extend the experiments to the multiple source domain adaptation scenario. We augment the pool of WSJ articles with articles coming from two other corpora: Genia and Brown. We want to gauge the effectiveness of the domain similarity measures in the multi-domain setting, where articles are selected from the pool of data without knowing their identity (which corpus the articles came from).

The test sets are the standard evaluation sets from the three corpora: the standard WSJ (section 23) and Brown test set from CoNLL 2008 (they contain 2,399 and 426 sentences, respectively) and the Genia test set (1,360 sentences). The mean sentence length is, respectively: 24.04, 16.8 and 26.2 words. As a reference, we give results of models trained on the respective corpora (*per-corpus* models; i.e. if we consider corpus boundaries and train a model on the respective domain – this model is 'supervised' in the sense that it knows from which corpus the test article came from) as well as a baseline model trained on all data, i.e. the *union* of all three corpora (WSJ+Genia+Brown), which is a standard baseline for domain adaptation techniques (e.g. Daumé III, 2007; McClosky et al., 2010).

	WSJ (38k)	**Brown** (28k)	**Genia** (19k)
random	86.58	73.81	83.77
per-corpus	87.50	81.55	86.63
union	87.05	79.12	81.57
topic model (var)	87.11⋆	81.76◇	86.77◇
words (js)	86.30	81.47◇	86.44◇

Table 7.6: Domain adaptation results on English (significantly better: ⋆ than random; ◇ than random and union).

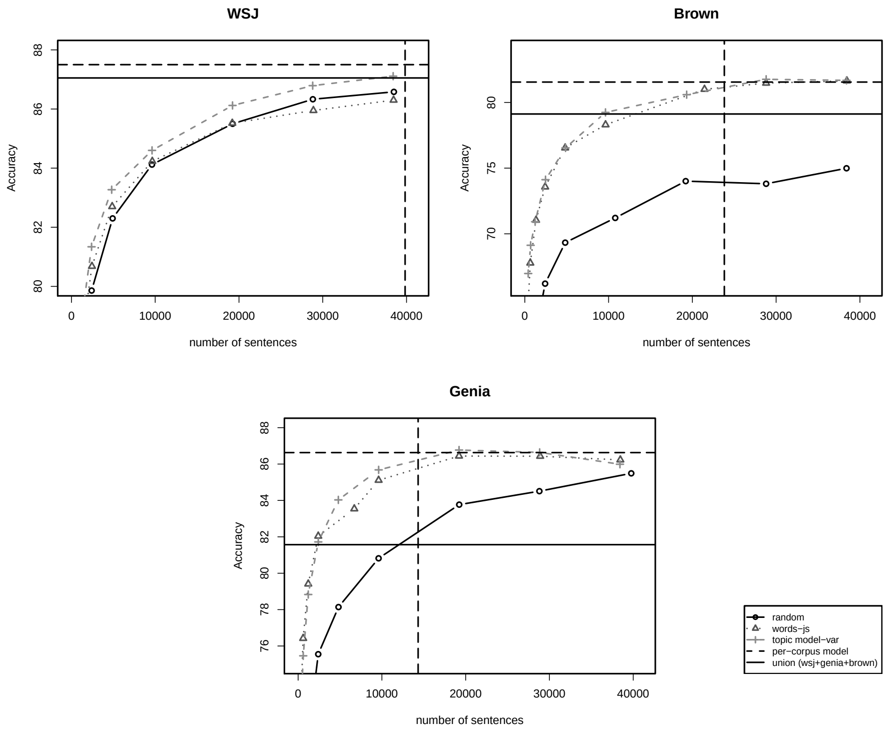


Figure 7.6: Domain adaptation results for English with increasing amounts of training data. The vertical line represents the amount of data the per-corpus model is trained on.

The learning curves are shown in Figure 7.6, while the scores for a specific amount of data are given in Table 7.6. The performance of the reference models (per-corpus and union) are indicated in Figure 7.6 with horizontal lines: the long-dashed line represents the per-corpus performance ('supervised' model); the solid line shows the performance of the union baseline trained on all available data (77k sentences). For the former, the vertical dashed lines indicate the amount of data the model was trained on (e.g. 23k sentences for Brown).

Simply taking all available data has a deteriorating effect: on all three test sets, the performance of the union model is below the presumably best performance of a model trained on the respective corpus (per-corpus model).

The empirical results show that automatic data selection by topic model outperforms random selection on all three test sets and the union baseline in two out of three cases. More specifically, selection by topic model outperforms random selection significantly on all three test sets and all points in the graph ($p < 0.05$). Selection by the word-based measure (words-js) achieves a significant improvement over the random baseline on two out of the three test sets – it falls below the random baseline on the WSJ test set. Thus, selection by topic model performs best – it achieves better performance than the union baseline with comparatively little data (Genia: 4k; Brown: 19k – in comparison: union has 77k). Moreover, it comes very close to the supervised per-corpus model performance[16] with a similar amount of data (cf. vertical dashed line). This is a very good result, given that the technique disregards the origin of the articles and just uses plain words as information. It automatically finds data that is beneficial for an unknown target domain.

Composition of the selected training data To give an idea about the data selected by the two automatic methods, Table 7.7 shows the composition of the training data for the models given in Table 7.6. As can be seen, the automatic measures always identified the largest proportion of articles from the presumably best origin (the corpus the test article stems from). For WSJ, topic model selected 98% WSJ articles, while the word-based measure selected 78% WSJ and 22% Brown data. This might be the reason for the performance difference between the two methods on the WSJ test data (cf. Figure 7.6).

A look at the Topic Model We will now examine the most probable words per topic distribution of the topic model used in the experiments. Moreover, we will compare it to an alternative topic model that is estimated on the collection of documents with stopwords removed. Note that the term *topic model* might be somewhat misleading here as a model that includes high-frequency words might capture more than topical information alone. As discussed in the following, very frequent words (closed class words that include stopwords) mark *style* quite well as is known from work on authorship attribution, cf. the foreword of the 3rd edition of Mosteller and Wallace (1964) by John Nerbonne.

[16]On Genia and Brown (cf. Table 7.6) there is no significant difference between topic model and per-corpus model.

	Target	**Sentences**	**Articles**	**WSJ**	**Brown**	**Genia**
words	WSJ	38454	580	78% (454)	22% (126)	0% (0)
(js)	Brown	28805	222	21% (49)	79% (174)	0% (0)
	Genia	19206	1223	1% (17)	4% (49)	95% (1157)
	Target	**Sentences**	**Articles**	**WSJ**	**Brown**	**Genia**
topic	WSJ	38406	1275	98% (1246)	2% (28)	0% (1)
model	Brown	28809	319	45% (145)	54% (174)	0% (0)
(var)	Genia	19211	1728	9% (165)	1% (13)	90% (1550)

Table 7.7: Composition of selected training data for models whose performance is given in Table 7.6.

Table 7.8 illustrates the most probable words of the topic model (that includes stopwords) estimated from the articles in the Genia, Brown and WSJ. Topic 1 and 37 seem to be related to the Genia domain (biomedical articles), while topic 36 reflects more the business domain (WSJ). In contrast, topic 0 mostly captures stopwords.

topic 0	the of a and in to for an is this with by we are ebp mutations mutation
topic 1	stat of gamma beta ifn in by cells alpha interleukin induced cytokine gene production th expression
...	
topic 36	president chief executive chairman officer mr was vice company and will named board years director old
topic 37	in of receptor patients glucocorticoid with receptors were and was binding gr dexamethasone normal not cortisol lymphocytes plasma mononuclear

Table 7.8: An illustration of the most probable words per topic (excerpt of topics out of 100) of the topic model estimated from the articles in the Genia, Brown and WSJ, which is used in the experiments.

As we have discussed previously, note that we did not remove stopwords in our experiments as we found no deteriorating effect while keeping them. To show this, Figure 7.7 compares the performance of selection by topic model

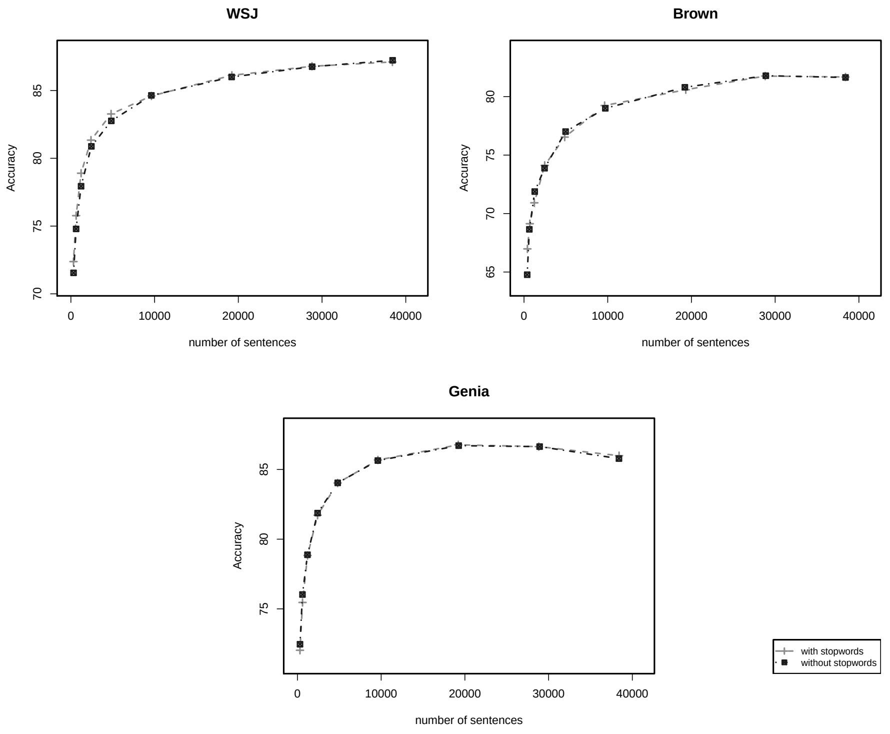


Figure 7.7: Comparison of the performance of selection by topic model with and without stopwords – there is so little difference between them that it is difficult to distinguish the curves.

with and without stopwords. Table 7.9 gives the most probable words of the topic model where stopwords were removed.

The topics in the latter model (Table 7.9) seems to be 'cleaner' in an information retrieval (IR) sense than those in Table 7.8 (the model we used). In IR, stopwords are usually neglected, as the interest is in finding documents that are related to other documents in terms of topic (content), while stopwords are often function words and as such are not considered to carry topical infor-

topic 0	activation kinase protein phosphorylation signaling tyrosine activated receptor signal cell induced activity pathway pathways pkc mediated dependent transduction ras
topic 1	state national abortion rights president issue political public anti majority election decision conservative democratic debate congress position issues states
topic 2	man eyes face head looked back turned water thought stood boy voice hands felt hand knew moment dark body

Table 7.9: An illustration of topics (out of 100) of a topic model estimated from the articles in the Genia, Brown and WSJ when stopwords were removed.

mation. Interestingly, on the contrary, function words are frequently used in authorship attribution, as it was found that they are useful for distinguishing one author from another (Mosteller & Wallace, 1964). Moreover, note that the closest work to ours (McClosky et al., 2010) did not remove stopwords either before calculating cosine similarity between corpora. Thus, obviously, the 50 most frequent words used in their study contain many stopwords – they are listed in McClosky (2010, chapter 5). They found that increasing the threshold to e.g. top 5000 words (and thus including more non-function words) was less effective than just keeping the top 50 (David McClosky, personal communication). This is an interesting issue, as it seems that modeling the distribution of these high-frequency words has a value. Maybe these words indeed include additional information that is related to style rather than topic, and that this is another factor to consider. However, it is not yet clear in how far they are important. In our case, including them did not hurt performance, while McClosky et al. (2010) kept them and obtained improved results. Thus, this remains an interesting open issue.

Combining measures Selection by topic model and words seem to capture somewhat different information, since the overlap between the two measures is less than 50% (cf. Table 7.4). Therefore, we examined a simple combination: select half of the training data by a measure based on words and the other half by topic model selection (while taking care to avoid duplicates).

Table 7.10 shows the results of combining the two similarity measures using this simple technique. We tested the combination of the two previously best-performing techniques (words using Jensen-Shannon and topic models using variational) as well as combinations between feature representations having low overlap: words with Jensen-Shannon and topic model using Euclidean (to have a metric from both the probabilistic and geometrical family) and words and topic model using Euclidean for both, as this combination had the overall lowest overlap (cf. Table 7.4).

This simple combination technique did not result in improved results. The performance of the combined model lies mostly between topic model and words, but did not outperform the best technique based on topic models. The combination based on Euclidean distance (for both representations) did not perform well, e.g. scored considerably below the random baseline on Genia.

	WSJ (38k)	**Brown** (28k)	**Genia** (19k)
random	86.58	73.81	83.77
tm (var)	87.11	81.76	86.77
words (js)	86.30	81.47	86.44
words (js)+tm (var)	86.63	81.45	86.60
words (js)+tm (euc)	86.49	81.40	86.30
words (euc)+tm (euc)	86.72	81.37	82.36

Table 7.10: Results of the simple combination technique that selects half of the training data by selection based on words and the other half by topic model.

Character tetragrams As mentioned earlier, we found character tetragrams less stable and hence did not discuss them in the empirical results section so far. To illustrate it, this section provides additional results for measures based on character tetragrams.

As shown in Figure 7.8 and Table 7.11, character tetragrams perform usually worse than topic model and word distributions. On the WSJ and Brown test sets, results are still somewhat close to their word-based counterparts, while on Genia the performance difference is large. On Genia, selection by character tetragrams performed considerably worse than random selection. By inspecting the composition of the training data (Table 7.12) we found that for Genia the character-based technique only selected 54%-60% of the articles from the Genia corpus, compared to the high 90-ies for the other techniques (cf. Table 7.7). Thus, character tetragrams did not perform that well.

As shown recently by Petrenz and Webber (2011), character tetragrams are working well for genre classification only as long as the genre-topic distribu-

	WSJ (38k)	**Brown** (28k)	**Genia** (19k)
random	86.58	73.81	83.77
tm (var)	87.11	81.76	86.77
words (js)	86.30	81.47	86.44
cn (var)	86.01	81.14	81.78
cn (js)	85.91	81.26	81.78

Table 7.11: Results including character tetragrams (cn).

tion remains stable, i.e., character tetragrams "owe their good results to the strong correlation between topics and genres." (Petrenz & Webber, 2011). At this point, we may only speculate on why character tetragrams did not perform well in our setting: maybe they are more affected by shifts in the genre-topic distribution than their word-based counterparts. But this would need further investigation.

	Target	**Sentences**	**Articles**	**WSJ**	**Brown**	**Genia**
char (var)	WSJ	38484	481	68% (328)	31% (152)	0% (1)
	Brown	28911	216	16% (34)	84% (182)	0% (0)
	Genia	19231	371	25% (91)	21% (79)	54% (201)
	Target	**Sentences**	**Articles**	**WSJ**	**Brown**	**Genia**
char (var)	WSJ	38408	459	65% (298)	35% (160)	0% (1)
	Brown	28862	215	14% (31)	86% (184)	0% (0)
	Genia	19211	389	18% (71)	22% (87)	60% (231)

Table 7.12: Composition of training data using character tetragrams.

So far we examined domain similarity measures for parsing, and concluded that selection by topic model performs best, closely followed by word-based selection using the Jensen-Shannon divergence. The question that remains is whether the measure is more widely applicable: How does it perform on another language and task?

PoS tagging To evaluate data selection with the automatic measures on another task, we perform similar domain adaptation experiments on WSJ, Genia and Brown for PoS tagging. We use two taggers (HMM and MaxEnt) and the

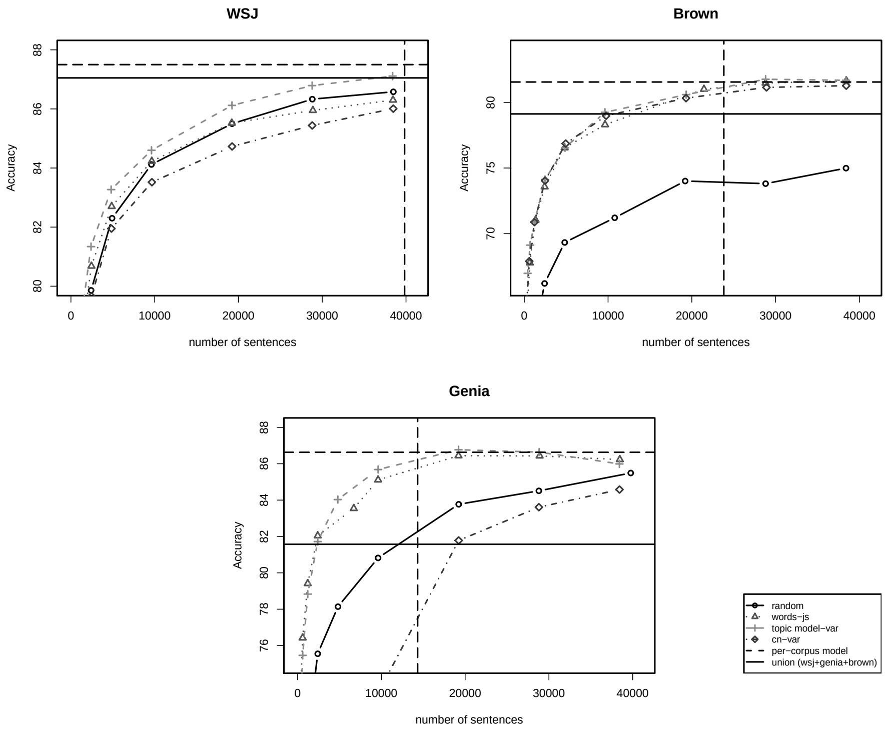
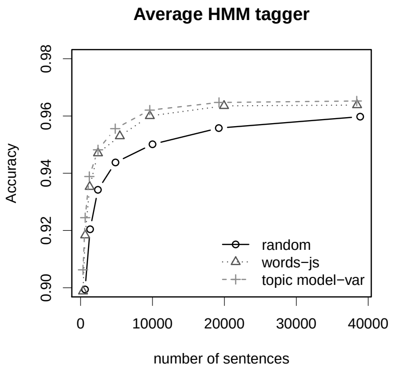


Figure 7.8: Results including character tetragrams (cn-var).

same three test articles as before. Figure 7.9 depicts the average over the three test sets, WSJ, Genia, Brown, for both taggers. The left figure shows the performance of the HMM tagger; on the right is the MaxEnt tagger.

The graphs show that automatic training data selection outperforms random data selection, and again topic model selection performs best closely followed by words using Jensen-Shannon. This confirms previous findings and shows that the domain similarity measures are effective also for this task.



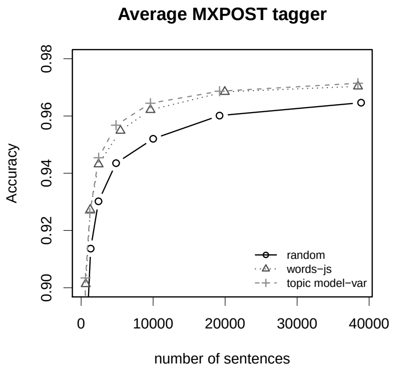


Figure 7.9: PoS tagging results for the two taggers, respectively, as average over 3 test sets.

7.6 Experiments on Dutch

For Dutch, we evaluate the approach on a bigger and more varied data set. It contains in total over 50k articles and 20 million words (cf. Table 7.2). In contrast to the English data, only a small portion of the data set is manually annotated: 281 articles (namely, the 186 DPC and 95 Wikipedia articles that were also used in Chapter 6, cf. Table 6.1).

Since we want to evaluate the performance of different similarity measures, we want to keep the influence of noise as low as possible. Therefore, we annotated the remaining articles with a parsing system that is more accurate (Plank & van Noord, 2010b), the Alpino parser. Note that using a more accurate parsing system to train another parser has recently also been proposed by Petrov, Chang, Ringgaard and Alshawi (2010) as *uptraining*. Alpino is a parser tailored to Dutch, that has been developed over the last ten years, and reaches an accuracy level of 90% on general newspaper text. It uses a conditional maximum entropy model as parse selection component. Details of Alpino are given in Chapter 2, Section 2.4.1.

7.6.1 Data and Results

For Dutch, we use articles from a variety of sources: Wikipedia,[17] EMEA[18] (documents from the European Medicines Agency) and the Dutch Parallel Corpus (DPC)[19] that covers a variety of subdomains.

The Dutch articles were parsed with Alpino and automatically converted to CoNLL format with the treebank conversion software from CoNLL 2006 (introduced in Chapter 6), where PoS tags have been replaced with the Alpino tags as that had a positive effect on the MST parser (cf. previous chapter). The 281 annotated articles come from all three sources. As with English, we consider as test set articles with at least 50 sentences, from which 30 are randomly sampled. The results on Dutch are shown in Figure 7.10. Domain similarity measures clearly outperform random data selection also in this setting with another language and a considerably larger pool of data (20 million words; 51k articles).

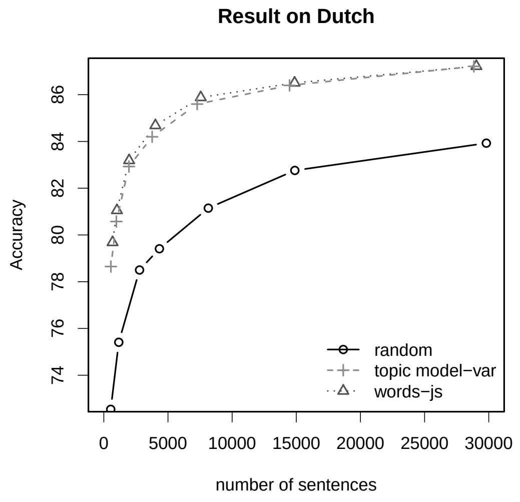


Figure 7.10: Result on Dutch: Parsing accuracy for increasing amounts of training data as average over 30 articles.

[17]http://ilps.science.uva.nl/WikiXML/

[18]http://urd.let.rug.nl/tiedeman/OPUS/EMEA.php

[19]http://www.kuleuven-kortrijk.be/DPC

7.7 Discussion

In this chapter we have shown the effectiveness of a simple technique that considers only plain words as domain selection measure for two tasks, dependency parsing and PoS tagging. Interestingly, human-annotated labels did not perform better than the automatic measure. The best technique is based on topic models and compares document topic distributions estimated by LDA (Blei et al., 2003) using the variational metric (very similar results were obtained using Jensen-Shannon). Topic model selection significantly outperforms random data selection on both examined languages, English and Dutch, and has a positive effect on PoS tagging. Moreover, it outperformed a standard domain adaptation baseline that trains a model on all available data (union) on two out of three test sets. Thus, less (but more domain-specific) data is often more fruitful than lots of unrelated data.

The performance of topic model selection is closely followed by that of the word-based measure using Jensen-Shannon divergence. By examining the overlap between the word-based and topic model-based technique, we found that despite similar performance their overlap is rather small. Given these results and the fact that no optimization has been done for the topic model itself, results are encouraging: there might be an even better measure that exploits the information from both techniques. So far, we tested a simple combination of the two by selecting half of the articles by a measure based on words and the other half by a measure based on topic models while testing different metrics. However, this simple combination technique did not improve results yet – topic model alone still performed best.

Overall, plain surface characteristics seem to carry important information about the domain of the text. Undoubtedly, parsing accuracy will be influenced by more factors than lexical information alone. Nevertheless, as we have seen, lexical differences constitute an important factor.

Venues for future research include: applying divergence measures over syntactic patterns (e.g. PoS n-gram sequences, dependency triples), adding additional articles to the pool of data by uptraining (Petrov et al., 2010), self-training (McClosky et al., 2006) or active learning (Hwa, 2004), gauging the effect of weighting instances according to their similarity to the test data (Jiang & Zhai, 2007; Plank & Sima'an, 2008), investigating further character tetragrams, automatic data shift detection (Dredze et al., 2010), comparing automatic selection to selection using Wikipedia categories (cf. Chapter 5), as well as analyzing differences between the gathered data.

Chapter 8

Summary and Conclusions

In this dissertation, we focused on the problem of the domain dependence of natural language parsing systems. We investigated techniques of domain adaptation that try to allay this problem. The first two chapters (part I) set the theoretical background, defined the task of parsing, introduced the parsing systems used in this work, and provided an introduction to domain adaptation by focusing on natural language parsing.

The aim of part II (that covers Chapter 4 and Chapter 5) was to investigate domain adaptation techniques in order to try to adapt the syntactic disambiguation model of the grammar-driven parser Alpino to new domains. Chapter 4 focused on supervised domain adaptation, the scenario in which some limited amount of labeled target data is available. We proposed the use of auxiliary distributions to integrate knowledge from a more general, out-of-domain model into a domain-specific model, for which only a very limited amount of labeled training data was available. We compared the approach to a range of straightforward baselines and prior work. In the next chapter (Chapter 5), we examined two main approaches to exploit unlabeled data: self-training and structural correspondence learning (SCL). The latter has had considerable impact in the field. Currently – as of the time of writing this dissertation – Blitzer's two papers on SCL (Blitzer et al., 2006, 2007) are the most cited papers on domain adaptation, they were cited 244 and 194 times according to Google Scholar. It was generally believed to be a promising domain adaptation technique.

However, the empirical results presented in part II of this dissertation are disappointing. Chapter 4 showed that the auxiliary-based approach that integrates knowledge from the more general model did not work for domain adaptation. Integrating the general model into the domain-specific model did not improve the parsing performance over the in-domain baseline that trains a model on the limited amount of in-domain training data only. To estimate the weights of the auxiliary features we need training data. When only small amounts of training data are available, the weights cannot be estimated appropriately and hence the contribution of these features is low when compared to the other features. Thus, the resulting model just achieved in-domain baseline accuracy. This result was confirmed on the two domains examined, questions and spoken data. However, both techniques from prior work, namely reference distribution and easy adapt, performed even worse. The reference distribution-based approach turned out to be the worst performing technique, indicating that relying too much on the out-of-domain model has a deteriorating effect on model performance. This would suggest that we should prefer the use of the auxiliary distribution-based approach, despite its empirical weakness. We conclude that whenever there is a (sometimes even small) amount of training data available for the target domain it is better to either just use that (i.e. apply the in-domain model) or adopt very simple adaptation techniques such as model combination (i.e. to train two models and linearly interpolate them).

In Chapter 5, we investigated unsupervised domain adaptation methods, self-training and structural correspondence learning (SCL). SCL has been applied successfully to PoS tagging and sentiment analysis, but was rather unexplored for parsing. We applied SCL to the parse disambiguation task and compared it to various instantiations of self-training. However, despite promising initial experiments and the general belief that SCL should be a valuable direction to explore, results were disappointing. SCL achieved only a minor improvement on two out of three test cases. This was due to improvements obtained on a few sentences only and the differences were not significant. Self-training performed even worse. None of the evaluated self-training variants (single versus multiple iterations, various selection techniques) worked: performance either improved only slightly, or degraded considerably most of the time. This would favor the use of the more complex technique (SCL) as it did not degrade performance as much as self-training. However, applying SCL involves many design choices and practical issues (amongst which the most crucial is the selection of pivot features). Thus, it is not really worth applying it to parse disambiguation. Therefore, the answer to our first research question on how effective domain adaptation techniques are, is negative. None of the proposed or investigated domain adaptation techniques (supervised or unsupervised) helped to considerably (and consistently) improve over baseline performance. So far, it is best to just use the available labeled data or apply very simple adaptation techniques like model combination.

Adapting the disambiguation component turned out to be a very hard task. Therefore, Chapter 6 presented an empirical investigation of the sensitivity of different parsing systems to domain shifts. To the best of our knowledge, no previous study has addressed this issue. We also proposed a simple measure to quantify domain sensitivity. In this chapter, we evaluated the variation in parsing accuracy of the grammar-driven system Alpino and two data-driven parsers, MST and Malt. We wanted to test the hypothesis that the grammar-driven system Alpino is less affected by domain shifts, in comparison to purely data-driven statistical parsing systems. Indeed, the chapter showed that the grammar-driven system Alpino is rather robust across domains. Alpino is the best performing system and it is significantly more robust than MST. In contrast, the transition-based parser Malt scores the lowest across all domains, but its variation turned out not to be different from Alpino. Overall, MST is the most domain-sensitive parser. This became apparent when we excluded lexical information from the training process, which resulted in a considerable performance drop for MST. Thus, with regard to our second research question on the domain sensitivity of various parsing systems, Chapter 6 showed that the grammar-driven parser Alpino suffered less from domain shifts, compared to the data-driven parsers, especially MST. Of course, these results are specific for Dutch; however, it is a first step. The proposed evaluation (and measure) is independent of both language and parsing system. In future it would be interesting to apply it to other languages or systems for which annotated data is available.

Most previous work on domain adaptation relied on the implicit assumption that domains are somehow given. That is, relevant (labeled or unlabeled) data was available, which has usually been determined by a human. In fact, we also assumed this in the first part of this dissertation. We either had target domain at our disposal or we exploited the category system of Wikipedia to find related target data. However, as more and more data becomes available, automatic ways to select data that is beneficial for a new (unknown) target domain are becoming attractive. Therefore, the goal of Chapter 7 was to evaluate measures of domain similarity to automatically acquire related training data for a given test set. The results showed that an unsupervised technique based on topic models is effective – it outperforms random data selection on both languages examined, English and Dutch. It is closely followed by a similarity measure based on words only (relative frequencies of words). Both automatic methods worked better than selection based on manually assigned labels gathered from meta-data that is available for English. Moreover, these automatic measures often outperformed a standard baseline model trained on the union of all data. Thus, simple relative frequencies of words turned out to be good indicator of domain. They were effective for training data selection, and less (but more domain-specific) data is often more fruitful than using all available data.

There are, however, still many untouched and open issues to explore. In Chapter 7, we selected increasing amounts of related target data but did not specify a stopping criterion. Therefore, one direction of future work is to establish such a criterion. This is related to the issue of *domain shift detection* (Dredze et al., 2010). They proposed a method to detect shifts in online data streams. Such a mechanism could indeed be used to trigger adaptation techniques, which is a second (related) direction we would like to explore. So far, domain adaptation studies assumed that a domain shift has occurred. Thirdly, we currently assume a single coherent document as target domain. If we are going to parse an entire corpus, it might be sensible to first aggregate similar documents and then try to find related target data. Other issues that we leave for future work include: exploring data selection in case the data was labeled with the parser that we want to improve (a kind of 'targeted' self-training – we only examined uptraining (Petrov et al., 2010), i.e. we used a more accurate parser to annotate data for another parser, which might not always be possible).

With regard to our last research question on the selection of training data, our empirical investigation in Chapter 7 showed that a simple measure based on word frequencies was effective for data selection. We believe that this is the case as the measure mostly captures lexical differences. It would be interesting to investigate more abstract feature representations that capture higher-level information (such as PoS tag sequences that have been used to capture more syntactic differences). On the other hand, using such a representation might again be problematic as the data will be noisy (PoS taggers are again trained on specific corpora and will suffer from domain shifts). In general, an interesting direction of future work would be to tease apart the different dimensions of variability between texts and examine their impact on parsing accuracy.

A large part of this dissertation focused on applying adaptation techniques to a parse disambiguation model. However, as shown in Chapter 6, the parsing system turned out to be rather robust across domains. This is not to say that domain dependence does not constitute a problem for grammar-driven parsers at all. However, the adaptation of other components, such as the lexicon, might be more fruitful. In future, we would like to investigate adaptation approaches for data-driven parsing systems. For example, instance weighting, data selection, and some very recently proposed approaches (e.g. Titov, 2011) that are based on SCL but do not require the selection of pivot features. It would also be interesting to evaluate the adaptation success in a subsequent task that relies on the parser's output (e.g. Sagae, 2010). Last but not least, as we have already mentioned, we believe that for the practical advance of parser portability the adaptation of pre-processing tools like PoS taggers or tokenizers will be necessary.

Part V

Appendix

Appendix A

Derivation of the Maximum Entropy Parametric Form

In this section, we derive the general parametric form of the conditional maximum entropy (MaxEnt) model as given in equation (2.13) in Chapter 2, restated here for convenience:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{m} \lambda_i f_i(x,y)\right) \tag{A.1}$$

The goal is to find the model $p(y|x)$ such that the conditional entropy of the model is maximized:

$$p_* = \arg\max_{p \in P} H(p) \tag{A.2}$$

where $H(p)$ is given by (Berger et al., 1996):

$$H(p) = -\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x) \tag{A.3}$$

We want to find the maximum of the above function while fulfilling two requirements: (a) the constraints given by the training data:

$$\begin{aligned} E_p[f_i] &= E_{\tilde{p}}[f_i] && \text{(A.4)} \\ \sum_{x,y} \tilde{p}(x) p(y|x) f_i(x,y) &= \sum_{x,y} \tilde{p}(x,y) f_i(x,y) && \text{(A.5)} \end{aligned}$$

and (b) the implicit requirement that $p(y|x)$ has to be a valid probability distribution:

$$\sum_{y \in \Omega(x)} p(y|x) = 1 \tag{A.6}$$

We can restate these constraints as:

$$\forall i, E_p[f_i] - E_{\tilde{p}}[f_i] = 0 \tag{A.7}$$

$$\sum_y p(y|x) - 1 = 0 \tag{A.8}$$

This is a problem of constraint optimization. To solve it, we can use the Lagrange multiplier technique to transform it into an unconstrained optimization problem. A parameter λ_i (a Lagrange multiplier) is introduced for each of the $m + 1$ constraints. That is, there is one Lagrange multiplier for every feature expectation constraint and one multiplier (λ_0, which we will denote by γ) for the requirement to get a proper probability distribution. The Lagrangian function (objective function plus lambda times constraints) is defined by:

$$\Lambda(p, \lambda) = H(p) + \sum_i \lambda_i (E_p[f_i] - E_{\tilde{p}}[f_i]) + \gamma \sum_y (p(y|x) - 1) \tag{A.9}$$

$$\begin{aligned}\Lambda(p, \lambda) = [&\overbrace{-\sum_{x,y} \tilde{p}(x) p(y|x) \log p(y|x)}^{H(p)} \\ &+ \sum_i \lambda_i (\sum_{x,y} \tilde{p}(x) p(y|x) f_i(x, y) - \sum_{x,y} \tilde{p}(x, y) f_i(x, y)) \\ &+ \gamma \sum_y (p(y|x) - 1)]\end{aligned} \tag{A.10}$$

We need to find the derivative of $\Lambda(p, \lambda)$ with respect to $p(y|x)$ and set it to zero, to derive the parametric form. Since the derivative of a sum is the sum of the derivatives of its terms, we can take the derivatives of the three terms one at a time.

For the first term, let's apply the product rule $(f \times g)' = f' \times g + f \times g')$:

$$\begin{aligned}
H(p) &= -[\overbrace{\tilde{p}(x)p(y|x)}^{f}\overbrace{\log p(y|x)}^{g}] && \text{(A.11)}\\
\frac{\partial H(p)}{\partial p(y|x)} &= -[\frac{\partial}{\partial p(y|x)}[\tilde{p}(x)p(y|x)]\log p(y|x) && \text{(A.12)}\\
&+ \tilde{p}(x)p(y|x)\frac{\partial}{\partial p(y|x)}\log p(y|x)] &&\\
&= -[\tilde{p}(x)\log p(y|x) + \tilde{p}(x)p(y|x)\frac{1}{p(y|x)}] && \text{(A.13)}\\
&= -[\tilde{p}(x)\log p(y|x) + \tilde{p}(x)] && \text{(A.14)}\\
&= -[\tilde{p}(x)(1+\log p(y|x))] && \text{(A.15)}
\end{aligned}$$

Next, we are looking for the partial derivative of the second term:

$$\sum_{x,y}\sum_{i}\lambda_i\tilde{p}(x)p(y|x)f_i(x,y) - \sum_{i}\lambda_i\tilde{p}(x,y)f_i(x,y) \qquad \text{(A.16)}$$

Since the second part corresponding to $E_{\tilde{p}}[f_i]$ does not contain $p(y|x)$, it falls away. We thus need to find the partial derivative of the first term, which is:

$$\frac{\partial}{\partial p(y|x)} = \sum_{i}\tilde{p}(x)\lambda_i f_i(x,y) \qquad \text{(A.17)}$$

The last term can be rewritten as:

$$\gamma\sum_{y}(p(y|x)-1) = \sum_{y}(\gamma p(y|x)-\gamma) \qquad \text{(A.18)}$$

The partial derivative of the last term with respect to $p(y|x)$ is:

$$\frac{\partial}{\partial p(y|x)} = \gamma \qquad \text{(A.19)}$$

Putting equation (A.15), (A.17) and (A.19) together we get:

$$\begin{aligned}
\frac{\partial\Lambda(p,\lambda)}{\partial p(y|x)} &= -[\tilde{p}(x)(1+\log p(y|x))] + \sum_{i}\tilde{p}(x)\lambda_i f_i(x,y) + \gamma && \text{(A.20)}\\
&= -\tilde{p}(x)(1+\log p(y|x)) + \tilde{p}(x)\sum_{i}\lambda_i f_i(x,y) + \gamma && \text{(A.21)}
\end{aligned}$$

Setting this equal to 0 and solving for $p(y|x)$:

$$0 = -\tilde{p}(x)(1 + \log p(y|x)) + \tilde{p}(x) \sum_i \lambda_i f_i(x,y) + \gamma \tag{A.22}$$

$$1 + \log p(y|x) = \frac{\tilde{p}(x) \sum_i \lambda_i f_i(x,y) + \gamma}{\tilde{p}(x)} \tag{A.23}$$

$$1 + \log p(y|x) = \sum_i \lambda_i f_i(x,y) + \frac{\gamma}{\tilde{p}(x)} \tag{A.24}$$

$$\log p(y|x) = \sum_i \lambda_i f_i(x,y) + \frac{\gamma}{\tilde{p}(x)} - 1 \tag{A.25}$$

$$p(y|x) = \exp(\sum_i \lambda_i f_i(x,y)) \exp(\frac{\gamma}{\tilde{p}(x)} - 1) \tag{A.26}$$

We can now substitute $p(y|x)$ from (A.26) into $\sum_y p(y|x) = 1$:

$$\sum_y \exp(\sum_i \lambda_i f_i(x,y)) \exp(\frac{\gamma}{\tilde{p}(x)} - 1) = 1 \tag{A.27}$$

$$\exp(\frac{\gamma}{\tilde{p}(x)} - 1) = \frac{1}{\sum_y \exp(\sum_i \lambda_i f_i(x,y))} \tag{A.28}$$

Let's call the denominator in (A.28) Z. Then:

$$\exp(\frac{\gamma}{\tilde{p}(x)} - 1) = \frac{1}{Z} \tag{A.29}$$

Now lets replace $\exp(\gamma/\tilde{p}(x) - 1)$ in equation (A.26) with the right-hand side of equation (A.29):

$$p(y|x) = \exp(\sum_i \lambda_i f_i(x,y)) \exp(\frac{\gamma}{\tilde{p}(x)} - 1) \tag{A.30}$$

$$= \exp(\sum_i \lambda_i f_i(x,y)) \frac{1}{Z} \tag{A.31}$$

Which gives us the parametric form of the conditional maximum entropy model:

$$p(y|x) = \frac{1}{Z} \exp(\sum_i \lambda_i f_i(x,y)) \tag{A.32}$$

with

$$Z = \sum_y \exp(\sum_i \lambda_i f_i(x,y))$$

Bibliography

Abney, S. (1997). Stochastic Attribute-Value Grammars. *Computational Linguistics*, *23*, 597–618.

Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. Chapman & Hall.

Ando, R. K. (2006). Applying Alternating Structure Optimization to Word Sense Disambiguation. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X).* New York City.

Ando, R. K. & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Berger, A., Della Pietra, S. & Della Pietra, V. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, *22*(1), 39-72.

Berger, A. & Printz, H. (1998). A Comparison of Criteria for Maximum Entropy / Minimum Divergence Feature Selection. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing* (pp. 97–106). Granada, Spain.

Biber, D. (1988). *Variation across speech and writing.* Cambridge University Press.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Blitzer, J. (2008). *Domain Adaptation of Natural Language Processing Systems*. Ph.d. thesis, University of Pennsylvania.

Blitzer, J., Dredze, M. & Pereira, F. (2007). Biographies, Bollywood, Boomboxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics.* Prague, Czech Republic.

Blitzer, J., McDonald, R. & Pereira, F. (2006). Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.* Sydney, Australia.

Bouma, G., van Noord, G. & Malouf, R. (2001). Alpino: Wide-coverage Computational Analysis of Dutch. In *Computational Linguistics in the Netherlands 2000.* Enschede.

Buchholz, S. & Marsi, E. (2006). CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)* (pp. 149–164). New York City.

Cahill, A., Burke, M., O'Donovan, R., Riezler, S., Genabith, J. van & Way, A. (2008). Wide-Coverage Deep Statistical Parsing using Automatic Dependency Structure Annotation. *Computational Linguistics*, *34*, 81–124.

Carroll, J. (2000). Statistical Parsing. In R. Dale, H. Moisl & H. Somers (Eds.), *Handbook of Natural Language Processing* (pp. 525–543). Marcel Dekker.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41-75.

Chang, M.-W., Connor, M. & Roth, D. (2010). The Necessity of Combining Adaptation Methods. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 767–777). Cambridge, MA: Association for Computational Linguistics.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)* (p. 1031-1036). Portland, Oregon.

Charniak, E. (1997). Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI)* (p. 598-603). Providence, Rhode Island.

Charniak, E. & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics.* Ann Arbor, Michigan: Association of Computational Linguistics.

Chelba, C. & Acero, A. (2006). Adaptation of Maximum Entropy Capitalizer: Little Data Can Help a Lot. *Computer Speech & Language*, *20*(4), 382–399.

Chen, S. & Rosenfeld, R. (1999). *A Gaussian Prior for Smoothing Maximum Entropy Models* (Tech. Rep.). Carnegie Mellon University, Pittsburg.

Chen, S. & Rosenfeld, R. (2000). A Survey of Smoothing Techniques for Maximum Entropy Models. *IEEE Transactions on Speech and Audio Processing*, *8*, 37–50.

Chen, W., Wu, Y. & Isahara, H. (2008). Learning Reliable Information for Dependency Parsing Adaptation. In *Proceedings of the 22nd International Conference on Computational Linguistics.* Manchester, UK.

Clark, S. (2010). Statistical Parsing. In A. Clark, C. Fox & S. Lappin (Eds.), *The*

Handbook of Computational Linguistics and Natural Language Processing (pp. 333–363). Wiley-Blackwell.

Collins, M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the ACL* (pp. 16–23). Madrid, Spain: Association for Computational Linguistics.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.d. thesis, University of Pennsylvania.

Daelemans, W., Zavrel, J., Berck, P. & Gillis, S. (1996). MBT: A Memory-Based Part of Speech Tagger-Generator. In *Proceedings of the Fourth Workshop on Very Large Corpora* (pp. 14–27). Kopenhagen.

Dagan, I., Lee, L. & Pereira, F. C. N. (1999). Similarity-Based Models of Word Cooccurrence Probabilities. *Machine Learning*, *34*, 43–69.

Daumé III, H. (2007). Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics.* Prague, Czech Republic: Association for Computational Linguistics.

Daumé III, H., Kumar, A. & Saha, A. (2010). Frustratingly Easy Semi-Supervised Domain Adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing* (pp. 53–59). Uppsala, Sweden: Association for Computational Linguistics.

Daumé III, H. & Marcu, D. (2006). Domain Adaptation for Statistical Classifiers. *Journal of Artificial Intelligence Research (JAIR)*, *26*, 101–126.

Dredze, M., Blitzer, J., Pratim Talukdar, P., Ganchev, K., Graca, J. & Pereira, F. (2007). Frustratingly Hard Domain Adaptation for Parsing. In *Proceedings of the 14th Conference on Natural Language Learning.* Prague, Czech Republic.

Dredze, M., Oates, T. & Piatko, C. (2010). We're Not in Kansas Anymore: Detecting Domain Changes in Streams. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 585–595). Cambridge, Massachusetts: Association for Computational Linguistics.

Finkel, J. R. & Manning, C. D. (2009). Hierarchical Bayesian Domain Adaptation. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 602–610). Boulder, Colorado: Association for Computational Linguistics.

Flickinger, D. (2000). On Building a More Efficient Grammar by Exploiting Types. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Processing with HPSG)*, 15–28.

Foster, J., Wagner, J., Seddah, D. & Genabith, J. van. (2007). Adapting WSJ-Trained Parsers to the British National Corpus using In-Domain Self-Training. In *Proceedings of the Tenth International Conference on Parsing*

Technologies (pp. 33–35). Prague, Czech Republic: Association for Computational Linguistics.

Gildea, D. (2001). Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing.* Pittsburgh, PA.

Goldberg, Y. & Elhadad, M. (2010). Inspecting the Structural Biases of Dependency Parsing Algorithms. In *Proceedings of the 14th Conference on Natural Language Learning* (pp. 234–242). Uppsala, Sweden: Association for Computational Linguistics.

Hara, T., Miyao, Y. & Tsujii, J. (2005). Adapting a Probabilistic Disambiguation Model of an HPSG Parser to a New Domain. In R. Dale, K.-F. Wong, J. Su & O. Y. Kwong (Eds.), *Natural Language Processing IJCNLP 2005* (Vol. 3651, p. 199-210). Springer Berlin / Heidelberg.

Hoekstra, H., Moortgat, M., Renmans, B., Schouppe, M., Schuurman, I. & Wouden, T. van der. (2003). *CGN syntactische annotatie.*

Hwa, R. (2004). Sample Selection for Statistical Parsing. *Computational Linguistics*, *30*, 253–276.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *The Physical Review*, *106*(4), 620–630.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press.

Jiang, J. (2008). *Domain Adaptation in Natural Language Processing*. Ph.d. thesis, University of Illinois at Urbana-Champaign, Urbana, Illionois.

Jiang, J. & Zhai, C. (2007). Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (pp. 264–271). Prague, Czech Republic: Association for Computational Linguistics.

Johansson, R. & Nugues, P. (2007). Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA.* Tartu, Estonia.

Johnson, M. (1998). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, *24*(4), 613-632.

Johnson, M., Geman, S., Canon, S., Chi, Z. & Riezler, S. (1999). Estimators for Stochastic "Unification-based" Grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics.* Maryland, USA: Association for Computational Linguistics.

Johnson, M. & Riezler, S. (2000). Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st Conference of the North American Chapter of the ACL* (pp. 154–161). Seattle, Washington.

Jurafsky, D. & Martin, J. H. (2008). *Speech and Language Processing* (2nd ed.). Prentice Hall.

Karlgren, J. & Cutting, D. (1994). Recognizing Text Genres with Simple Metrics using Discriminant Analysis. In *Proceedings of the 15th International Conference on Computational Linguistics* (pp. 1071–1075). Kyoto, Japan: Association for Computational Linguistics.

Kifer, D., Ben-David, S. & Gehrke, J. (2004). Detecting Change in Data Streams. In *Proceedings of the 30th International Conference on Very Large Data Bases* (p. 180-191).

Kilgarriff, A. & Rose, T. (1998). Measures for corpus similarity and homogeneity. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing* (pp. 46–52). ACL-SIGDAT.

Kübler, S., McDonald, R. & Nivre, J. (2009). *Dependency Parsing*. Morgan and Claypool publishers.

Lease, M. & Charniak, E. (2005). Parsing Biomedical Literature. In *In Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05), Jeju Island, Korea* (pp. 58–69).

Lease, M., Charniak, E., Johnson, M. & McClosky, D. (2006). A look at parsing and its applications. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence and Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI).* Boston, Massachusetts.

Lee, D. Y. (2001). Genres, registers, text types, domains and styles: clarifying the concepts and navigating a path through the BNC jungle. *Technology, 5*, 37–72.

Lee, L. (1997). *Similarity-Based Approaches to Natural Language Processing*. Ph.d. thesis, Harvard University, Cambridge, MA.

Lee, L. (2001). On the Effectiveness of the Skew Divergence for Statistical Language Analysis. In *In Artificial Intelligence and Statistics 2001* (pp. 65–72). Key West, Florida.

Lee, L. (2004). "I'm sorry Dave, I'm afraid I can't do that": Linguistics, Statistics, and Natural Language Processing circa 2001. In C. on the Fundamentals of Computer Science: Challenges, C. S. Opportunities & N. R. C. Telecommunications Board (Eds.), *Computer Science: Reflections on the Field, Reflections from the Field* (pp. 111–118). The National Academies Press.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *Information Theory, IEEE Transactions on*, *37*(1), 145 -151.

Lippincott, T., Ó Séaghdha, D., Sun, L. & Korhonen, A. (2010). Exploring variation across biomedical subdomains. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 689–697). Beijing, China.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning.* Taipei.

Malouf, R. (2010). Maximum Entropy Models. In A. Clark, C. Fox & S. Lappin (Eds.), *The Handbook of Computational Linguistics and Natural Language Processing* (pp. 131–154). Wiley-Blackwell.

Malouf, R. & van Noord, G. (2004). Wide Coverage Parsing with Stochastic Attribute Value Grammars. In *Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses.* Hainan China.

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press, Cambridge Mass.

McClosky, D. (2010). *Any Domain Parsing: Automatic Domain Adaptation for Parsing*. Ph.d. thesis, Brown University.

McClosky, D. & Charniak, E. (2008). Self-Training for Biomedical Parsing. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics* (pp. 101–104). Columbus, Ohio: Association for Computational Linguistics.

McClosky, D., Charniak, E. & Johnson, M. (2006). Effective Self-Training for Parsing. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 152–159). New York: Association for Computational Linguistics.

McClosky, D., Charniak, E. & Johnson, M. (2010). Automatic Domain Adaptation for Parsing. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 28–36). Los Angeles, California: Association for Computational Linguistics.

McDonald, R. & Nivre, J. (2007). Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 122–131). Prague, Czech Republic.

McDonald, R., Pereira, F., Ribarov, K. & Hajič, J. (2005). Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing* (pp. 523–530). Vancouver, British Columbia, Canada: Association for Computational Linguistics.

Moore, R. C. & Lewis, W. (2010). Intelligent Selection of Language Model Training Data. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics* (pp. 220–224). Uppsala, Sweden: Association for Computational Linguistics.

Mosteller, F. & Wallace, D. (1964). *Inference and Disputed Authorship: The Federalist Papers*. Series in Behavioral Science: Quantitative Methods, Addison Wesley.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G. ülsen, Kübler, S. et al. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, *13*, 95–135.

Nivre, J. & McDonald, R. (2008). Integrating Graph-Based and Transition-Based Dependency Parsers. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics* (pp. 950–958). Columbus, Ohio: Association for Computational Linguistics.

Noreen, E. W. (1989). *Computer-intensive methods for testing hypotheses: An introduction*. Wiley-Interscience. Paperback.

Oostdijk, N. (2000). The Spoken Dutch Corpus: Overview and first evaluation. In *Proceedings of second international conference on language resources and evaluation (lrec)* (p. 887-894). Athens, Greece.

Osborne, M. (2000). Estimation of stochastic attribute-value grammars using an informative sample. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics* (pp. 586–592). Saarbrücken, Germany: Association for Computational Linguistics.

Pan, S. J. & Yang, Q. (2010, October). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359.

Petrenz, P. & Webber, B. (2011). Stable Classification of Text Genres. *Computational Linguistics*, *37*(2), 385-393.

Petrov, S., Chang, P.-C., Ringgaard, M. & Alshawi, H. (2010). Uptraining for Accurate Deterministic Question Parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 705–713). Cambridge, MA: Association for Computational Linguistics.

Plank, B. (2009a). A Comparison of Structural Correspondence Learning and Self-training for Discriminative Parse Selection. In *Proceedings of the NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing* (pp. 37–42). Boulder, Colorado, USA: Association for Computational Linguistics.

Plank, B. (2009b). Structural Correspondence Learning for Parse Disambiguation. In *Proceedings of the Student Research Workshop at EACL 2009* (pp. 37–45). Athens, Greece: Association for Computational Linguistics.

Plank, B. & Sima'an, K. (2008). Subdomain Sensitive Statistical Parsing using Raw Corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation.* Marrakech, Morocco.

Plank, B. & van Noord, G. (2008). Exploring an Auxiliary Distribution Based Approach to Domain Adaptation of a Syntactic Disambiguation Model. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation* (pp. 9–16). Manchester, UK: Coling 2008 Organizing Committee.

Plank, B. & van Noord, G. (2010a). Dutch Dependency Parser Performance Across Domains. In E. Westerhout, T. Markus & P. Monachesi (Eds.), *Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands* (pp. 123–138). Utrecht, The Netherlands: LOT Occasional Series.

Plank, B. & van Noord, G. (2010b). Grammar-driven versus Data-driven: Which Parsing System Is More Affected by Domain Shifts? In *Proceedings of the ACL Workshop on NLP and Linguistics: Finding the Common Ground* (pp. 25–33). Uppsala, Sweden: Association for Computational Linguistics.

Plank, B. & van Noord, G. (2011). Effective Measures of Domain Similarity for Parsing. In *Proceedings of the 49th Meeting of the Association for Computational Linguistics* (pp. 1566–1576). Portland, Oregon, USA: Association for Computational Linguistics.

Prins, R. & van Noord, G. (2003). Reinforcing Parser Preferences through Tagging. *Traitement Automatique des Langues*, *44*(3), 121–139.

Ratnaparkhi, A. (1998). *Maximum entropy models for natural language ambiguity resolution*. Ph.d. thesis, University of Pennsylvania.

Ratnaparkhi, A. (1999). Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning*, *34*(1-3), 151–175.

Ravi, S., Knight, K. & Soricut, R. (2008). Automatic Prediction of Parser Accuracy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* (pp. 887–896). Honolulu, Hawaii: Association for Computational Linguistics.

Reichart, R. & Rappoport, A. (2007). Self-Training for Enhancement and Domain Adaptation of Statistical Parsers Trained on Small Datasets. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (pp. 616–623). Prague, Czech Republic: Association for Computational Linguistics.

Rényi, A. (1961). On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability* (pp. 547–561). Berkeley.

Riezler, S. & Maxwell, J. T. (2005). On Some Pitfalls in Automatic Evaluation and Significance Testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (pp. 57–64). Ann Arbor, Michigan: Association for Computational Linguistics.

Rimell, L. & Clark, S. (2008). Adapting a Lexicalized-Grammar Parser to Contrasting Domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* (pp. 475–484). Honolulu, Hawaii: Association for Computational Linguistics.

Roark, B. & Bacchiani, M. (2003). Supervised and unsupervised PCFG adaptation to novel domains. In *Proceedings of the 4th Conference of the North American Chapter of the ACL* (pp. 126–133). Edmonton, Canada: Association for Computational Linguistics.

Roland, D. & Jurafsky, D. (1998). How verb subcategorization frequencies are affected by corpus choice. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics* (pp. 1122–1128). Montreal, Quebec, Canada: Association for Computational Linguistics.

Sagae, K. (2010). Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the ACL Workshop on Domain Adaptation for Natural Language Processing* (pp. 37–44). Uppsala, Sweden: Association for Computational Linguistics.

Sagae, K., Miyao, Y. & Tsujii, J. (2007). HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (pp. 624–631). Prague, Czech Republic: Association for Computational Linguistics.

Sagae, K. & Tsujii, J. (2007). Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles. In *Proceedings of the 7th Conference on Natural Language Learning* (pp. 1044–1050). Prague, Czech Republic: Association for Computational Linguistics.

Sancho, L., De La Torre, R., Gerda Horneck, G., Ascaso, C., De Los Rios, A., Pintado, A. et al. (2007). Lichens survive in space: results from the 2005 LICHENS experiment. *Astrobiology*, *7*(3), 443–454.

Santorini, B. (1990). *Part-Of-Speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing)* (Tech. Rep.). Philadelphia, PA, USA: Department of Linguistics, University of Pennsylvania.

Sekine, S. (1997). The Domain Dependence of Parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* (pp. 96–102). Washington D.C., USA.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar* (Vol. 4). Stanford, CA: Center for the Study of Language and Information.

Shimizu, N. & Nakagawa, H. (2007). Structural Correspondence Learning for Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007.* Prague, Czech Republic.

Steedman, M., Osborne, M., Sarkar, A., Clark, S., Hwa, R., Hockenmaier, J. et al. (2003). Bootstrapping Statistical Parsers from Small Datasets. In *Proceedings of the 11th Conference of the European Chapter of the ACL* (pp. 331–338). Budapest, Hungary: Association for Computational Linguistics.

Steyvers, M. & Griffiths, T. (2007). Probabilistic Topic Models. In T. Landauer,

D. Mcnamara, S. Dennis & W. Kintsch (Eds.), *Handbook of Latent Semantic Analysis.* Lawrence Erlbaum Associates. Hardcover.

Surdeanu, M. & Manning, C. D. (2010). Ensemble Models for Dependency Parsing: Cheap and Good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 649–652). Los Angeles, California: Association for Computational Linguistics.

Titov, I. (2011). Domain Adaptation by Constraining Inter-Domain Variability of Latent Feature Representation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 62–71). Portland, Oregon, USA: Association for Computational Linguistics.

Van Asch, V. & Daelemans, W. (2010). Using Domain Similarity for Performance Estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing* (pp. 31–36). Uppsala, Sweden: Association for Computational Linguistics.

van Noord, G. (2006). At Last Parsing Is Now Operational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles* (pp. 20–42). Leuven.

van Noord, G. (2007). Using Self-Trained Bilexical Preferences to Improve Disambiguation Accuracy. In *Proceedings of the 10th International Workshop on Parsing Technologies* (pp. 1–10). Prague.

van Noord, G. (2009). Learning Efficient Parsing. In (pp. 817–825). Athens, Greece: Association for Computational Linguistics.

van Noord, G. & Malouf, R. (2005). *Wide Coverage Parsing with Stochastic Attribute Value Grammars.* (Draft. A preliminary version of this paper was published in the Proceedings of the IJCNLP workshop Beyond Shallow Analyses, Hainan China, 2004.)

Velldal, E. & Oepen, S. (2005). Maximum Entropy Models for Realization Ranking. In *Proceedings of MT-Summit X.* Phuket, Thailand.

Webber, B. (2009). Genre distinctions for Discourse in the Penn TreeBank. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (pp. 674–682). Suntec, Singapore: Association for Computational Linguistics.

Wu, Z., Markert, K. & Sharoff, S. (2010). Fine-Grained Genre Classification Using Structural Learning Algorithms. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics* (pp. 749–759). Uppsala, Sweden: Association for Computational Linguistics.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 38th Meeting of the Association for Computational*

Linguistics (pp. 947–953). Saarbrücken, Germany: Association for Computational Linguistics.

Zhang, Y. & Wang, R. (2009). Cross-Domain Dependency Parsing Using a Deep Linguistic Grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (pp. 378–386). Suntec, Singapore: Association for Computational Linguistics.

Zhu, X. & Goldberg, A. B. (2009). *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers.

Nederlandse samenvatting

Het doel van de computationele taalkunde is het maken van systemen die in staat zijn natuurlijke taal te begrijpen en te produceren, net zoals wij mensen dat doen. Het maken van dergelijke systemen moeilijk, onder andere vanwege het probleem van de ambiguïteit van natuurlijke taal.

In dit proefschrift ligt de focus op het automatisch ontleden. Voor een gegeven uiting wordt bepaald welke woorden en woordgroepen bij elkaar horen, en wat de functie van de verschillende woordgroepen is (zoals onderwerp, lijdend voorwerp, of bepaling). Om uiteindelijk de betekenis van een zin te kunnen achterhalen, is de syntactische ontleding een belangrijke eerste stap. Het probleem van ambiguïteit doet zich ook hier voor. In een zin als 'Zulke boeken lezen de mensen nu eenmaal graag' is het voor een computer niet onmiddellijk evident dat 'zulke boeken' het lijdend voorwerp is, en 'de mensen' het onderwerp. Een computer zal wellicht net zo goed kunnen kiezen voor de absurde omgekeerde ontleding.

Om zulke problemen van ambiguïteit aan te pakken wordt veelal gebruik gemaakt van machinaal leren. Hierbij worden technieken ingezet waarmee computers een model leren op basis van een grote hoeveelheid voorbeelden. In het geval van automatisch ontleden wordt een grote hoeveelheid zinnen handmatig van de juiste syntactische structuur voorzien. Vervolgens leert de computer aan de hand van deze voorbeelden (de *trainingdata*) een model. Zo'n model wordt geleerd door parameters te berekenen op basis van soms wel duizenden kenmerken van de trainingdata. Een zo'n kenmerk is bijvoorbeeld: het onderwerp van de zin gaat wel of niet aan de persoonsvorm vooraf. Het systeem leert dan, bijvoorbeeld, dat een ontleding waarbij het onderwerp aan de persoonsvorm vooraf gaat de voorkeur verdient.

Het blijkt dat deze aanpak voor het automatisch ontleden goede resultaten geeft zolang de trainingdata representatief is. Dus, als de trainingdata bestaat uit krantenartikelen uit de Volkskrant dan zal het resulterende model vooral goede prestaties halen op andere krantenartikelen uit de Volkskrant. Maar als we dat model toepassen op een wetenschappelijk essay over oceanografie, of de notulen van een afdelingsvergadering, of een redevoering van een afgevaardigde in het Europees parlement, dan worden de prestaties snel veel minder.

Automatische ontleedsystemen zijn dus sterk afhankelijk van het *domein* van de teksten waaruit de trainingdata is opgebouwd: waar gaan de teksten over, wat is de stijl van de teksten, wat is het genre, en zo verder. Dit is het probleem van domeinafhankelijkheid. Dit is een algemeen probleem van machinaal leren, dat pas in de afgelopen jaren aandacht begint te krijgen.

Dit proefschrift onderzoekt de domeinafhankelijkheid van automatische ontleedsystemen. De trainingdata voor een automatisch ontleedsysteem bestaat uit een verzameling van syntactisch geannoteerde zinnen. Zulke data bestaat alleen voor specifieke domeinen en de hoeveelheid is beperkt, omdat het annoteren van de data handmatig moet gebeuren. Zo zijn de meeste huidige ontleedsystemen getraind op krantentekst, en de toepassing op andere domeinen werkt dus minder goed. Het doel van het onderzoek is het ontwikkelen van technieken die de automatische aanpassing van taalverwerkingssystemen voor nieuwe domeinen mogelijk maken, zonder de noodzaak van het handmatig annoteren van nieuwe data.

De belangrijkste bijdragen van dit proefschrift zijn de volgende. Na een inleiding in het automatisch ontleden (hoofdstuk 2) en het probleem van domeinafhankelijkheid (hoofstuk 3) onderzoeken we in het tweede deel van dit proefschrift (in de hoofdstuken 4 en 5) de effectiviteit van nieuwe en bestaande algoritmes voor het aanpassen van modellen aan andere domeinen. Deze worden geëvalueerd in het kader van een automatisch ontleedsysteem voor het Nederlands dat gebaseerd is op een handgeschreven grammatica, de Alpino parser. Deze experimenten zijn nieuw, want eerder werk was gericht op domeinafhankelijkheid van ontleedsystemen die in zijn geheel zijn gebaseerd op data (datagedreven ontleedsystemen). Hoofdstuk 4 onderzoekt een aantal technieken die ervan uit gaan dat er – naast een hoeveelheid algemene data – een kleine hoeveelheid geannoteerde data van het nieuwe domein ter beschikking staat. In dit scenario is dus voor het toepassen van een ontleedsysteem op een nieuw domein nog wel wat annotatiewerk vereist. Daarentegen onderzoekt hoofdstuk 5 algoritmes waarbij helemaal geen handmatig annotatiewerk vereist is. Hierbij wordt wel data uit het nieuwe domein gebruikt, maar voor die data zijn geen handmatig toegekende annotaties beschikbaar.

In het derde deel bekijken we de gevoeligheid van verschillende soorten ontleedsystemen op domeinverschuivingen. We vergelijken Alpino, het ontleedsysteem gebaseerd op een handgeschreven grammatica, met de prestaties van datagedreven systemen op verschillende soorten tekst. De hypothese dat het grammatica-gebaseerde systeem minder beïnvloed wordt door domeinverschuivingen wordt getest, en, dus, dat datagebaseerde systemen meer behoefte hebben aan technieken voor aanpassing aan nieuwe domeinen. Het hoofdstuk laat zien dat Alpino robust is in vergelijking met de ontleedsystemen die in zijn geheel gebaseerd zijn op geannoteerde data. Alpino wordt minder beïnvloed door domeinverschuivingen.

De laatste bijdrage van dit proefschrift is de ontwikkeling van een meetinstrument om aan te geven in hoeverre twee teksten tot een verschillend of juist tot een vergelijkbaar domein behoren. De meeste studies nemen aan dat er geannoteerde data dan wel data zonder annotatie van het nieuwe domein ter beschikking staat. Dit is echter niet altijd het geval. Daarom evalueren we in het vierde deel van het proefschrift maten om automatisch geschikte trainingdata te selecteren voor een nieuw domein. De resultaten tonen aan dat een eenvoudige techniek gebaseerd op distributies van woorden effectief is voor het selecteren van trainingdata voor beide onderzochte talen, het Engels en het Nederlands.

Groningen Dissertations in Linguistics (GRODIL)

25. Roel Jonkers (1998). *Comprehension and Production of Verbs in Aphasic Speakers.*
26. Erik F. Tjong Kim Sang (1998). *Machine Learning of Phonotactics.*
27. Paulien Rijkhoek (1998). *On Degree Phrases and Result Clauses.*
28. Jan de Jong (1999). *Specific Language Impairment in Dutch: Inflectional Morphology and Argument Structure.*
29. H. Wee (1999). *Definite Focus.*
30. Eun-Hee Lee (2000). *Dynamic and Stative Information in Temporal Reasoning: Korean Tense and Aspect in Discourse.*
31. Ivilin Stoianov (2001). *Connectionist Lexical Processing.*
32. Klarien van der Linde (2001). *Sonority Substitutions.*
33. Monique Lamers (2001). *Sentence Processing: Using Syntactic, Semantic, and Thematic Information.*
34. Shalom Zuckerman (2001). *The Acquisition of "Optional" Movement.*
35. Rob Koeling (2001). *Dialogue-Based Disambiguation: Using Dialogue Status to Improve Speech Understanding.*
36. Esther Ruigendijk (2002). *Case Assignment in Agrammatism: a Cross-linguistic Study.*
37. Tony Mullen (2002). *An Investigation into Compositional Features and Feature Merging for Maximum Entropy-Based Parse Selection.*
38. Nanette Bienfait (2002). *Grammatica-onderwijs aan allochtone jongeren.*
39. Dirk-Bart den Ouden (2002). *Phonology in Aphasia: Syllables and Segments in Level-specific Deficits.*
40. Rienk Withaar (2002). *The Role of the Phonological Loop in Sentence Comprehension.*
41. Kim Sauter (2002). *Transfer and Access to Universal Grammar in Adult Second Language Acquisition.*
42. Laura Sabourin (2003). *Grammatical Gender and Second Language Processing: An ERP Study.*
43. Hein van Schie (2003). *Visual Semantics.*
44. Lilia Schürcks-Grozeva (2003). *Binding and Bulgarian.*
45. Stasinos Konstantopoulos (2003). *Using ILP to Learn Local Linguistic Structures.*
46. Wilbert Heeringa (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance.*
47. Wouter Jansen (2004). *Laryngeal Contrast and Phonetic Voicing: A Laboratory Phonology Approach to English, Hungarian and Dutch.*
48. Judith Rispens (2004). *Syntactic and Phonological Processing in Developmental Dyslexia.*
49. Danielle Bougaïré (2004). *L'approche communicative des campagnes de sensibilisation en santé publique au Burkina Faso: les cas de la planification familiale, du sida et de l'excision.*
50. Tanja Gaustad (2004). *Linguistic Knowledge and Word Sense Disambiguation.*
51. Susanne Schoof (2004). *An HPSG Account of Nonfinite Verbal Complements in Latin.*

52. M. Begoña Villada Moirón (2005). *Data-driven identification of fixed expressions and their modifiability.*
53. Robbert Prins (2005). *Finite-State Pre-Processing for Natural Language Analysis.*
54. Leonoor van der Beek (2005). *Topics in Corpus-Based Dutch Syntax.*
55. Keiko Yoshioka (2005). *Linguistic and gestural introduction and tracking of referents in L1 and L2 discourse.*
56. Sible Andringa (2005). *Form-focused instruction and the development of second language proficiency.*
57. Joanneke Prenger (2005). *Taal telt! Een onderzoek naar de rol van taalvaardigheid en tekstbegrip in het realistisch wiskundeonderwijs.*
58. Neslihan Kansu-Yetkiner (2006). *Blood, Shame and Fear: Self-Presentation Strategies of Turkish Women's Talk about their Health and Sexuality.*
59. Mónika Z. Zempléni (2006). *Functional imaging of the hemispheric contribution to language processing.*
60. Maartje Schreuder (2006). *Prosodic Processes in Language and Music.*
61. Hidetoshi Shiraishi (2006). *Topics in Nivkh Phonology.*
62. Tamás Biró (2006). *Finding the Right Words: Implementing Optimality Theory with Simulated Annealing.*
63. Dieuwke de Goede (2006). *Verbs in Spoken Sentence Processing: Unraveling the Activation Pattern of the Matrix Verb.*
64. Eleonora Rossi (2007). *Clitic production in Italian agrammatism.*
65. Holger Hopp (2007). *Ultimate Attainment at the Interfaces in Second Language Acquisition: Grammar and Processing.*
66. Gerlof Bouma (2008). *Starting a Sentence in Dutch: A corpus study of subject- and object-fronting.*
67. Julia Klitsch (2008). *Open your eyes and listen carefully. Auditory and audiovisual speech perception and the McGurk effect in Dutch speakers with and without aphasia.*
68. Janneke ter Beek (2008). *Restructuring and Infinitival Complements in Dutch.*
69. Jori Mur (2008). *Off-line Answer Extraction for Question Answering.*
70. Lonneke van der Plas (2008). *Automatic Lexico-Semantic Acquisition for Question Answering.*
71. Arjen Versloot (2008). *Mechanisms of Language Change: Vowel reduction in 15th century West Frisian.*
72. Ismail Fahmi (2009). *Automatic term and Relation Extraction for Medical Question Answering System.*
73. Tuba Yarbay Duman (2009). *Turkish Agrammatic Aphasia: Word Order, Time Reference and Case.*
74. Maria Trofimova (2009). *Case Assignment by Prepositions in Russian Aphasia.*
75. Rasmus Steinkrauss (2009). *Frequency and Function in WH Question Acquisition. A Usage-Based Case Study of German L1 Acquisition.*
76. Marjolein Deunk (2009). *Discourse Practices in Preschool. Young Children's Participation in Everyday Classroom Activities.*

77. Sake Jager (2009). *Towards ICT-Integrated Language Learning: Developing an Implementation Framework in terms of Pedagogy, Technology and Environment.*
78. Francisco Dellatorre Borges (2010). *Parse Selection with Support Vector Machines.*
79. Geoffrey Andogah (2010). *Geographically Constrained Information Retrieval.*
80. Jacqueline van Kruiningen (2010). *Onderwijsontwerp als conversatie. Probleemoplossing in interprofessioneel overleg.*
81. Robert G. Shackleton (2010). *Quantitative Assessment of English-American Speech Relationships.*
82. Tim Van de Cruys (2010). *Mining for Meaning: The Extraction of Lexico-semantic Knowledge from Text.*
83. Therese Leinonen (2010). *An Acoustic Analysis of Vowel Pronunciation in Swedish Dialects.*
84. Erik-Jan Smits (2010). *Acquiring Quantification. How Children Use Semantics and Pragmatics to Constrain Meaning.*
85. Tal Caspi (2010). *A Dynamic Perspective on Second Language Development.*
86. Teodora Mehotcheva (2010). *After the fiesta is over. Foreign language attrition of Spanish in Dutch and German Erasmus Students.*
87. Xiaoyan Xu (2010). *English language attrition and retention in Chinese and Dutch university students.*
88. Jelena Prokić (2010). *Families and Resemblances.*
89. Radek Šimík (2011). *Modal existential wh-constructions.*
90. Katrien Colman (2011). *Behavioral and neuroimaging studies on language processing in Dutch speakers with Parkinson's disease.*
91. Siti Mina Tamah (2011). *A Study on Student Interaction in the Implementation of the Jigsaw Technique in Language Teaching.*
92. Aletta Kwant (2011). *Geraakt door prentenboeken. Effecten van het gebruik van prentenboeken op de sociaal-emotionele ontwikkeling van kleuters.*
93. Marlies Kluck (2011). *Sentence amalgamation.*
94. Anja Schüppert (2011). *Origin of asymmetry: Mutual intelligibility of spoken Danish and Swedish.*
95. Peter Nabende (2011). *Applying Dynamic Bayesian Networks in Transliteration Detection and Generation.*
96. Barbara Plank (2011). *Domain Adaptation for Parsing.*
97. Çağrı Çöltekin (2011). *Catching Words in a Stream of Speech: Computational simulations of segmenting transcribed child-directed speech.*
98. Dörte Hessler (2011). *Audiovisual Processing in Aphasic and Non-Brain-Damaged Listeners: The Whole is More than the Sum of its Parts.*

GRODIL
Secretary of the Department of General Linguistics
Postbus 716
9700 AS Groningen
The Netherlands