

Chapter 1

Statistical Parsing

1.1	Introduction	1
1.2	Basic Concepts and Terminology	2
1.3	Probabilistic Context-Free Grammars	7
1.4	Generative Models	11
1.5	Discriminative Models	18
1.6	Beyond Supervised Parsing	26
1.7	Summary and Conclusions	29
	Acknowledgments	31
	Bibliography	31

This chapter describes techniques for statistical parsing, that is, methods for syntactic analysis that make use of statistical inference from samples of natural language text. The major topics covered are probabilistic context-free grammars, supervised parsing using generative and discriminative models, and models for unsupervised parsing.

1.1 Introduction

By *statistical parsing* we mean techniques for syntactic analysis that are based on statistical inference from samples of natural language. Statistical inference may be invoked for different aspects of the parsing process but is primarily used as a technique for disambiguation, that is, for selecting the most appropriate analysis of a sentence from a larger set of possible analyses, for example, those licensed by a formal grammar. In this way, statistical parsing methods complement and extend the classical parsing algorithms for formal grammars described in the Syntactic Parsing Chapter.

The application of statistical methods to parsing started in the 1980s, drawing on work in the area of corpus linguistics, inspired by the success of statistical speech recognition, and motivated by some of the perceived weaknesses of parsing systems rooted in the generative linguistics tradition and based solely on hand-built grammars and disambiguation heuristics. In statistical parsing, these grammars and heuristics are wholly or partially replaced by statistical models induced from corpus data. By capturing distributional tendencies in the data, these models can rank competing analyses for a sentence, which fa-

cilitates *disambiguation*, and can therefore afford to impose fewer constraints on the language accepted, which increases *robustness*. Moreover, since models can be induced automatically from data, it is relatively easy to port systems to new languages and domains, as long as representative data sets are available.

Against this, however, it must be said that most of the models currently used in statistical parsing require data in the form of syntactically annotated sentences – a *treebank* – which can turn out to be quite a severe bottleneck in itself, in some ways even more severe than the old knowledge acquisition bottleneck associated with large-scale grammar development. Since the range of languages and domains for which treebanks are available is still limited, the investigation of methods for learning from unlabeled data, particularly when adapting a system to a new domain, is therefore an important problem on the current research agenda. Nevertheless, practically all high-precision parsing systems currently available are dependent on learning from treebank data, although often in combination with hand-built grammars or other independent resources. It is the models and techniques used in those systems that are the topic of this chapter.

The rest of the chapter is structured as follows. Section 1.2 introduces a conceptual framework for characterizing statistical parsing systems in terms of syntactic representations, statistical models, and algorithms for learning and inference. Section 1.3 is devoted to the framework of Probabilistic Context-Free Grammar (PCFG), which is arguably the most important model for statistical parsing, not only because it is widely used in itself but because some of its perceived limitations have played an important role in guiding the research towards improved models, discussed in the rest of the chapter. Section 1.4 is concerned with approaches that are based on generative statistical models, of which the PCFG model is a special case, and Section 1.5 discusses methods that instead make use of conditional or discriminative models. While the techniques reviewed in Sections 1.4–1.5 are mostly based on supervised learning, that is, learning from sentences labeled with their correct analyses, Section 1.6 is devoted to methods that start from unlabeled data, either alone or in combination with labeled data. Finally, in Section 1.7, we summarize and conclude.

1.2 Basic Concepts and Terminology

The task of a statistical parser is to map sentences in natural language to their preferred syntactic representations, either by providing a ranked list of candidate analyses or by selecting a single optimal analysis. Since the latter case can be regarded as a special case of the former (a list of length one), we will assume without loss of generality that the output is always a ranked

list. We will use \mathcal{X} for the set of possible inputs, where each input $x \in \mathcal{X}$ is assumed to be a sequence of tokens $x = w_1, \dots, w_n$, and we will use \mathcal{Y} for the set of possible syntactic representations. In other words, we will assume that the input to a parser comes pre-tokenized and segmented into sentences, and we refer to the Text Pre-Processing Chapter for the intricacies hidden in this assumption when dealing with raw text. Moreover, we will not deal directly with cases where the input does not take the form of a string, such as word-lattice parsing for speech recognition, even though many of the techniques covered in this chapter can be generalized to such cases.

1.2.1 Syntactic Representations

The set \mathcal{Y} of possible syntactic representations is usually defined by a particular theoretical framework or treebank annotation scheme but normally takes the form of a complex graph or tree structure. The most common type of representation is a *constituent structure* (or *phrase structure*), where a sentence is recursively decomposed into smaller segments that are categorized according to their internal structure into *noun phrases*, *verb phrases*, etc. Constituent structures are naturally induced by context-free grammars (Chomsky 1956) and are assumed in many theoretical frameworks of natural language syntax, for example, Lexical Functional Grammar (LFG) (Kaplan and Bresnan 1982; Bresnan 2000), Tree Adjoining Grammar (TAG) (Joshi 1985; Joshi 1997), and Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag 1987; Pollard and Sag 1994). They are also widely used in annotation schemes for treebanks, such as the Penn Treebank scheme for English (Marcus, Santorini, and Marcinkiewicz 1993; Marcus, Santorini, Marcinkiewicz, MacIntyre, Bies, Ferguson, Katz, and Schasberger 1994), and in the adaptations of this scheme that have been developed for Chinese (Xue, Xia, Chiou, and Palmer 2004), Korean (Han, Han, Ko, and Palmer 2002), Arabic (Maamouri and Bies 2004), and Spanish (Moreno, López, Sánchez, and Grishman 2003). Figure 1.1 shows a typical constituent structure for an English sentence, taken from the Wall Street Journal section of the Penn Treebank.

Another popular type of syntactic representation is a *dependency structure*, where a sentence is analyzed by connecting its words by binary asymmetrical relations called *dependencies*, and where words are categorized according to their functional role into *subject*, *object*, etc. Dependency structures are adopted in theoretical frameworks such as Functional Generative Description (Sgall, Hajičová, and Panevová 1986) and Meaning-Text Theory (Mel'čuk 1988) and are used for treebank annotation especially for languages with free or flexible word order. The best known dependency treebank is the Prague Dependency Treebank of Czech (Hajič, Vidova Hladká, Panevová, Hajičová, Sgall, and Pajas 2001; Böhmová, Hajič, Hajičová, and Hladká 2003), but dependency-based annotation schemes have been developed also for Arabic (Hajič, Smrž, Zemánek, Šnaidauf, and Beška), Basque (Aduriz, Aranzabe, Arriola, Atutxa, Díaz de Ilarraza, Garmendia, and Oronoz 2003), Danish

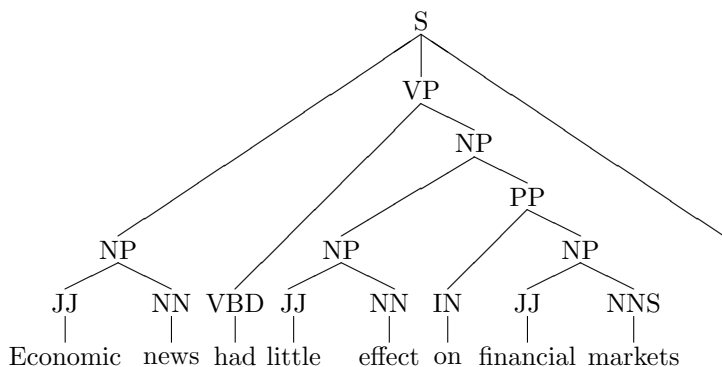


FIGURE 1.1: Constituent structure for an English sentence taken from the Penn Treebank.

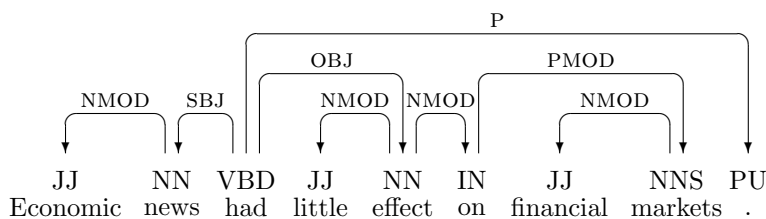


FIGURE 1.2: Dependency structure for an English sentence taken from the Penn Treebank.

(Kromann 2003), Greek (Prokopidis, Desypri, Koutsombogera, Papageorgiou, and Piperidis 2005), Russian (Boguslavsky, Grigorieva, Grigoriev, Kreidlin, and Frid 2000), Slovene (Džeroski, Erjavec, Ledinek, Pajas, Žabokrtsky, and Žele 2006), Turkish (Ofłazer, Say, Hakkani-Tür, and Tür 2003), and other languages. Figure 1.2 shows a typical dependency representation of the same sentence as in Figure 1.1.

A third kind of syntactic representation is found in *categorial grammar*, which connects syntactic (and semantic) analysis to inference in a logical calculus. The syntactic representations used in categorial grammar are essentially proof trees, which cannot be reduced to constituency or dependency representations, although they have affinities with both. In statistical parsing, categorial grammar is mainly represented by Combinatory Categorial Grammar (CCG) (Steedman 2000), which is also the framework used in CCGbank (Hockenmaier and Steedman 2007), a reannotation of the Wall Street Journal section of the Penn Treebank.

In most of this chapter, we will try to abstract away from the particular

representations used and concentrate on concepts of statistical parsing that cut across different frameworks, and we will make reference to different syntactic representations only when this is relevant. Thus, when we speak about assigning an analysis $y \in \mathcal{Y}$ to an input sentence $x \in \mathcal{X}$, it will be understood that the analysis is a syntactic representation as defined by the relevant framework.

1.2.2 Statistical Parsing Models

Conceptually, a statistical parsing model can be seen as consisting of two components:

1. A *generative* component GEN that maps an input x to a set of candidate analyses $\{y_1, \dots, y_k\}$, that is, $\text{GEN}(x) \subseteq \mathcal{Y}$ (for $x \in \mathcal{X}$).
2. An *evaluative* component EVAL that ranks candidate analyses via a numerical scoring scheme, that is, $\text{EVAL}(y) \in \mathbb{R}$ (for $y \in \text{GEN}(x)$).

Both the generative and the evaluative component may include parameters that need to be estimated from empirical data using statistical inference. This is the *learning* problem for a statistical parsing model, and the data set used for estimation is called the *training set*. Learning may be *supervised* or *unsupervised*, depending on whether sentences in the training set are labeled with their correct analyses or not (cf. the Fundamental Statistical Techniques Chapter). In addition, there are *weakly supervised* learning methods, which combine the use of labeled and unlabeled data.

The distinction between the generative and evaluative component of a statistical parsing model is related to, but not the same as, the distinction between generative and discriminative models (cf. the Fundamental Statistical Techniques Chapter). In our setting, a generative model is one that defines a joint probability distribution over inputs and outputs, i.e., that defines the probability $P(x, y)$ for any input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. By contrast, a discriminative model only makes use of the conditional probability of the output given the input, i.e., the probability $P(y|x)$. As a consequence, discriminative models are often used to implement the evaluative component of a complete parsing model, while generative models usually integrate the generative and the evaluative components into one model. However, as we shall see in later sections, there are a number of variations possible on this basic theme.

Given that a statistical parsing model has been learned from data, we need an efficient way of constructing and ranking the candidate analyses for a given input sentence. This is the *inference* problem for a statistical parser. Inference may be *exact* or *approximate*, depending on whether or not the inference algorithm is guaranteed to find the optimal solution according to the model. We shall see that there is often a tradeoff between having the advantage of a more complex model but needing to rely on approximate inference, on the

one hand, and adopting a more simplistic model but being able to use exact inference, on the other.

1.2.3 Parser Evaluation

The *accuracy* of a statistical parser, that is, the degree to which it succeeds in finding the preferred analysis for an input sentence, is usually evaluated by running the parser on a sample of sentences $X = \{x_1, \dots, x_m\}$ from a treebank, called the *test set*. Assuming that the treebank annotation y_i for each sentence $x_i \in X$ represents the preferred analysis, the *gold standard parse*, we can measure the *test set accuracy* of the parser by comparing its output $f(x_i)$ to the gold standard parse y_i , and we can use the test set accuracy to estimate the expected accuracy of the parser on sentences from the larger population represented by the test set.

The simplest way of measuring test set accuracy is to use the *exact match* metric, which simply counts the number of sentences for which the parser output is identical to the treebank annotation, that is, $f(x_i) = y_i$. This is a rather crude metric, since an error in the analysis of a single word or constituent has exactly the same impact on the result as the failure to produce any analysis whatsoever, and the most widely used evaluation metrics today are therefore based on various kinds of partial correspondence between the parser output and the gold standard parse.

For parsers that output constituent structures, the most well-known evaluation metrics are the PARSEVAL metrics (Black, Abney, Flickinger, Gdaniec, Grishman, Harrison, Hindle, Ingria, Jelinek, Klavans, Liberman, Roukos, Santorini, and Strzalkowski 1991; Grishman, Macleod, and Sterling 1992), which consider the number of matching constituents between the parser output and the gold standard. For dependency structures, the closest correspondent to these metrics is the *attachment score* (Buchholz and Marsi 2006), which measures the proportion of words in a sentence that are attached to the correct head according to the gold standard. Finally, to be able to compare parsers that use different syntactic representations, several researchers have proposed evaluation schemes where both the parser output and the gold standard parse are converted into sets of more abstract dependency relations, so-called *dependency banks* (Lin 1995; Lin 1998; Carroll, Briscoe, and Sanfilippo 1998; Carroll, Minnen, and Briscoe 2003; King, Crouch, Riezler, Dalrymple, and Kaplan 2003; Forst, Bertomeu, Crysmann, Fouvry, Hansen-Schirra, and Kordoni 2004).

The use of treebank data for parser evaluation is in principle independent of its use in parser development and is not limited to the evaluation of statistical parsing systems. However, the development of statistical parsers normally involves an iterative training-evaluation cycle, which makes statistical evaluation an integral part of the development. This gives rise to certain methodological issues, in particular the need to strictly separate data that are used for repeated testing during development – *development sets* – from data that

are used for evaluation of the final system – test sets.

It is important in this context to distinguish two different but related problems: *model selection* and *model assessment*. Model selection is the problem of estimating the performance of different models in order to choose the (approximate) best one, which can be achieved by testing on development sets or by cross-validation on the entire training set. Model assessment is the problem of estimating the expected accuracy of the finally selected model, which is what test sets are typically used for.

1.3 Probabilistic Context-Free Grammars

In the preceding section, we introduced the basic concepts and terminology that we need to characterize different models for statistical parsing, including methods for learning, inference and evaluation. We start our exploration of these models and methods in this section by examining the framework of probabilistic context-free grammar.

1.3.1 Basic Definitions

A *probabilistic context-free grammar* (PCFG) is a simple extension of a context-free grammar in which every production rule is associated with a probability (Booth and Thompson 1973). Formally, a PCFG is a quintuple $G = (\Sigma, N, S, R, D)$, where Σ is a finite set of terminal symbols, N is a finite set of non-terminal symbols (disjoint from Σ), $S \in N$ is the start symbol, R is a finite set of production rules of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$, and $D : R \rightarrow [0, 1]$ is a function that assigns a probability to each member of R (cf. the Syntactic Parsing Chapter on context-free grammars). Figure 1.3 shows a PCFG capable of generating the sentence in Figure 1.1 with its associated parse tree. Although the actual probabilities assigned to the different rules are completely unrealistic because of the very limited coverage of the grammar, it nevertheless serves to illustrate the basic form of a PCFG.

As usual, we use $L(G)$ to denote the string language generated by G , that is, the set of strings over the terminal alphabet Σ for which there exists a derivation $S \Rightarrow^* x$ using rules in R . In addition, we use $T(G)$ to denote the tree language generated by G , that is, the set of parse trees corresponding to valid derivations of strings in $L(G)$. Given a parse tree $y \in T(G)$, we use $\text{YIELD}(y)$ for the terminal string in $L(G)$ associated with y , $\text{COUNT}(i, y)$ for the number of times that the i th production rule $r_i \in R$ is used in the derivation of y , and $\text{LHS}(i)$ for the nonterminal symbol in the left-hand side of r_i .

The probability of a parse tree $y \in T(G)$ is defined as the product of

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
.	→	.	1.0	IN	→	on	1.0

FIGURE 1.3: Probabilistic context-free grammar for a fragment of English.

probabilities of all rule applications in the derivation of y :

$$P(y) = \prod_{i=1}^{|R|} D(r_i)^{\text{COUNT}(i,y)} \quad (1.1)$$

This follows from basic probability theory on the assumption that the application of a rule in the derivation of a tree is independent of all other rule applications in that tree, a rather drastic independence assumption that we will come back to. Since the yield of a parse tree uniquely determines the string associated with the tree, the joint probability of a tree $y \in T(G)$ and a string $x \in L(G)$ is either 0 or equal to the probability of y , depending on whether or not the string matches the yield:

$$P(x, y) = \begin{cases} P(y) & \text{if YIELD}(y) = x \\ 0 & \text{otherwise} \end{cases} \quad (1.2)$$

It follows that the probability of a string can be obtained by summing up the probabilities of all parse trees compatible with the string:

$$P(x) = \sum_{y \in T(G): \text{YIELD}(y)=x} P(y) \quad (1.3)$$

A PCFG is *proper* if P defines a proper probability distribution over every subset of rules that have the same left-hand side $A \in N$:¹

$$\sum_{r \in R: \text{LHS}(r)=A} D(r) = 1 \quad (1.4)$$

A PCFG is *consistent* if it defines a proper probability distribution over the set of trees that it generates:

$$\sum_{y \in T(G)} P(y) = 1 \quad (1.5)$$

¹The notion of properness is sometimes considered to be part of the definition of a PCFG, and the term *weighted* CFG (WCFG) is then used for a non-proper PCFG (Smith and Johnson 2007).

Consistency can also be defined in terms of the probability distribution over *strings* generated by the grammar. Given equation 1.3, the two notions are equivalent.

1.3.2 PCFGs as Statistical Parsing Models

PCFGs have many applications in natural language processing, for example, in language modeling for speech recognition or statistical machine translation, where they can be used to model the probability distribution of a string language. In this chapter, however, we are only interested in their use as statistical parsing models, which can be conceptualized as follows:

- The set \mathcal{X} of possible inputs is the set Σ^* of strings over the terminal alphabet, and the set \mathcal{Y} of syntactic representations is the set of all parse trees over Σ and N .
- The generative component is the underlying context-free grammar, that is, $\text{GEN}(x) = \{y \in T(G) \mid \text{YIELD}(x) = y\}$.
- The evaluative component is the probability distribution over parse trees, that is, $\text{EVAL}(y) = P(y)$.

For example, even the minimal PCFG in Figure 1.3 generates two trees for the sentence in Figure 1.1, the second of which is shown in Figure 1.4. According to the grammar, the probability of the parse tree in Figure 1.1 is 0.0000794, while the probability of the parse tree in Figure 1.4 is 0.0001871. In other words, using this PCFG for disambiguation, we would prefer the second analysis, which attaches the PP *on financial markets* to the verb *had*, rather than to the noun *effect*. According to the gold standard annotation in the Penn Treebank, this would not be the correct choice.

Note that the score $P(y)$ is equal to the joint probability $P(x, y)$ of the input sentence and the output tree, which means that a PCFG is a generative model (cf. the Fundamental Statistical Techniques Chapter). For evaluation in a parsing model, it may seem more natural to use the conditional probability $P(y|x)$ instead, since the sentence x is given as input to the model. The conditional probability can be derived as shown in Equation 1.6, but since the probability $P(x)$ is a constant normalizing factor, this will never change the internal ranking of analyses in $\text{GEN}(x)$.

$$P(y|x) = \frac{P(x, y)}{\sum_{y' \in \text{GEN}(x)} P(y')} \quad (1.6)$$

1.3.3 Learning and Inference

The *learning* problem for the PCFG model can be divided into two parts: learning a context-free grammar $G = (\Sigma, N, S, R)$, and learning the probability assignment D for rules in R . If a pre-existing context-free grammar is

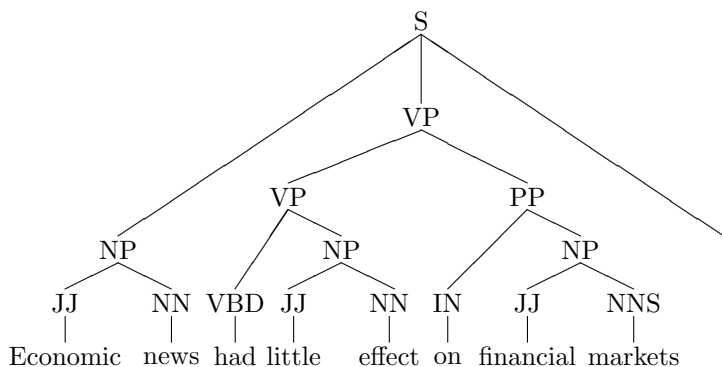


FIGURE 1.4: Alternative constituent structure for an English sentence taken from the Penn Treebank (cf. figure 1.1).

used, then only the rule probabilities need to be learned. Broadly speaking, learning is either supervised or unsupervised, depending on whether it presupposes that sentences in the training set are annotated with their preferred analysis.

The simplest method for supervised learning is to extract a so-called *treebank grammar* (Charniak 1996), where the context-free grammar contains all and only the symbols and rules needed to generate the trees in the training set $Y = \{y_1, \dots, y_m\}$, and where the probability of each rule is estimated by its relative frequency among rules with the same left-hand side:

$$D(r_i) = \frac{\sum_{j=1}^m \text{COUNT}(i, y_j)}{\sum_{j=1}^m \sum_{r_k \in R: \text{LHS}(r_k) = A} \text{COUNT}(k, y_j)} \quad (1.7)$$

To give a simple example, the grammar in Figure 1.3 is in fact a treebank grammar for the treebank consisting of the two trees in Figure 1.1 and Figure 1.4. The grammar contains exactly the rules needed to generate the two trees, and rule probabilities are estimated by the frequency of each rule relative to all the rules for the same nonterminal. Treebank grammars have a number of appealing properties. First of all, relative frequency estimation is a special case of maximum likelihood estimation (MLE), which is a well understood and widely used method in statistics (cf. the Fundamental Statistical Techniques Chapter). Secondly, treebank grammars are guaranteed to be both proper and consistent (Chi and Geman 1998). Finally, both learning and inference is simple and efficient. However, although early investigations reported encouraging results for treebank grammars, especially in combination with other statistical models (Charniak 1996; Charniak 1997), empirical research has clearly shown that they do not yield the most accurate parsing models, for reasons that we will return to in Section 1.4.

If treebank data are not available for learning, but the context-free grammar is given, then unsupervised methods for maximum likelihood estimation can be used to learn rules probabilities. The most commonly used method is the Inside-Outside algorithm (Baker 1979), which is a special case of Expectation-Maximization (EM), as described in the Fundamental Statistical Techniques Chapter. This algorithm was used in early work on PCFG parsing to estimate the probabilistic parameters of hand-crafted context-free grammars from raw text corpora (?; Pereira and Schabes 1992). Like treebank grammars, PCFGs induced by the Inside-Outside algorithm are guaranteed to be proper and consistent (Sánchez and Benedí 1997; Chi and Geman 1998). We will return to unsupervised learning for statistical parsing in Section 1.6.

The *inference* problem for the PCFG model is to compute, given a specific grammar G and an input sentence x , the set $\text{GEN}(x)$ of candidate representations and to score each candidate by the probability $P(y)$, as defined by the grammar. The first part is simply the parsing problem for CFGs, and many of the algorithms for this problem discussed in the Syntactic Parsing Chapter have a straightforward extension that computes the probabilities of parse trees in the same process. This is true, for example, of the CKY algorithm (Ney 1991), Earley’s algorithm (Stolcke 1995), and the algorithm for bilexical context-free grammars described in Eisner and Satta (1999) and Eisner (2000).

These algorithms are all based on dynamic programming, which makes it possible to compute the probability of a substructure at the time when it is being composed of smaller substructures and use Viterbi search to find the highest scoring analysis in $O(n^3)$ time, where n is the length of the input sentence. However, this also means that, although the model as such defines a complete ranking over all the candidate analyses in $\text{GEN}(x)$, these parsing algorithms only compute the single best analysis. Nevertheless, the inference is exact in the sense that the analysis returned by the parser is guaranteed to be the most probable analysis according to the model. There are generalizations of this scheme that instead extract the k best analyses, for some constant k , with varying effects on time complexity (Jiménez and Marzal 2000; Charniak and Johnson 2005; Huang and Chiang 2005). A comprehensive treatment of many of the algorithms used in PCFG parsing can be found in Goodman (1999).

1.4 Generative Models

Using a simple treebank grammar of the kind described in the preceding section to rank alternative analyses generally does not lead to very high parsing accuracy. The reason is that, because of the independence assumptions built

into the PCFG model, such a grammar does not capture the dependencies that are most important for disambiguation. In particular, the probability of a rule application is independent of the larger tree context in which it occurs.

This may mean, for example, that the probability with which a noun phrase is expanded into a single pronoun is constant for all structural contexts, even though it is a well-attested fact for many languages that this type of noun phrase is found more frequently in subject position than in object position. It may also mean that different verb phrase expansions (i.e., different configurations of complements and adjuncts) are generated independently of the lexical verb that functions as the syntactic head of the verb phrase, despite the fact that different verbs have different subcategorization requirements.

In addition to the lack of structural and lexical sensitivity, a problem with this model is that the children of a node are all generated in a single atomic event, which means that variants of the same structural realizations (e.g., the same complement in combination with different sets of adjuncts or even punctuation) are treated as disjoint events. Since the trees found in many treebanks tend to be rather flat, with a high average branching factor, this often leads to a very high number of distinct grammar rules with data sparseness as a consequence. In an often cited experiment, Charniak (1996) counted 10,605 rules in a treebank grammar extracted from a 300,000 word subset of the Penn Treebank, only 3943 of which occurred more than once.

These limitations of simple treebank PCFGs have been very important in guiding research on statistical parsing during the last ten to fifteen years, and many of the models proposed can be seen as targeting specific weaknesses of these simple generative models. In this section, we will consider techniques based on more complex generative models, with more adequate independence assumptions. In the next section, we will discuss approaches that abandon the generative paradigm in favor of conditional or discriminative models.

1.4.1 History-Based Models

One of the most influential approaches in statistical parsing is the use of a *history-based model*, where the derivation of a syntactic structure is modeled by a stochastic process and the different steps in the process are conditioned on events in the derivation history. The general form of such a model is the following:

$$P(y) = \prod_{i=1}^m P(d_i | \Phi(d_1, \dots, d_{i-1})) \quad (1.8)$$

where $D = d_1, \dots, d_m$ is a derivation of y and Φ is a function that defines which events in the history are taken into account in the model.² By way

²Note that the standard PCFG model can be seen as a special case of this, for example, by letting D be a leftmost derivation of y according to the context-free grammar and by letting $\Phi(d_1, \dots, d_{i-1})$ be the left-hand side of the production used in d_i .

of example, let us consider one of the three generative lexicalized models proposed by Collins (1997). In these models, nonterminals have the form $A(a)$, where A is an ordinary nonterminal label (such as NP or VP) and a is a terminal corresponding to the lexical head of A . In Model 2, the expansion of a node $A(a)$ is defined as follows:

1. Choose a head child H with probability $P_h(H|A, a)$.
2. Choose left and right subcat frames, LC and RC , with probabilities $P_{lc}(LC|A, H, h)$ and $P_{rc}(RC|A, H, h)$.
3. Generate the left and right modifiers (siblings of $H(a)$) $L_1(l_1), \dots, L_k(l_k)$ and $R_1(r_1), \dots, R_m(r_m)$ with probabilities $P_l(L_i, l_i|A, H, h, \delta(i-1), LC)$ and $P_r(R_i, r_i|A, H, h, \delta(i-1), RC)$.

In the third step, children are generated inside-out from the head, meaning that $L_1(l_1)$ and $R_1(r_1)$ are the children closest to the head child $H(a)$. Moreover, in order to guarantee a correct probability distribution, the farthest child from the head on each side is a dummy child labeled STOP. The subcat frames LC and RC are multisets of ordinary (non-lexicalized) nonterminals, and elements of these multisets get deleted as the corresponding children are generated. The distance metric $\delta(j)$ is a function of the surface string from the head word h to the outermost edge of the j th child on the same side, which returns a vector of three features: (1) Is the string of zero length? (2) Does the string contain a verb? (3) Does the string contain 0, 1, 2 or more than 2 commas?

To see what this means for a concrete example, consider the following phrase, occurring as part of an analysis for *Last week Marks bought Brooks*:

$$\begin{aligned}
 P(S(\text{bought}) \rightarrow \text{NP}(\text{week}) \text{NP-C}(\text{Marks}) \text{VP}(\text{bought})) = & \quad (1.9) \\
 & P_h(\text{VP}|S, \text{bought}) \times \\
 & P_{lc}(\{\text{NP-C}\}|S, \text{VP}, \text{bought}) \times \\
 & P_{rc}(\{\}|S, \text{VP}, \text{bought}) \times \\
 & P_l(\text{NP-C}(\text{Marks})|S, \text{VP}, \text{bought}, \langle 1, 0, 0 \rangle, \{\text{NP-C}\}) \times \\
 & P_l(\text{NP}(\text{week})|S, \text{VP}, \text{bought}, \langle 0, 0, 0 \rangle, \{\}) \times \\
 & P_l(\text{STOP}|S, \text{VP}, \text{bought}, \langle 0, 0, 0 \rangle, \{\}) \times \\
 & P_r(\text{STOP}|S, \text{VP}, \text{bought}, \langle 0, 0, 0 \rangle, \{\})
 \end{aligned}$$

This should be compared with the corresponding treebank PCFG, which has a single model parameter for the conditional probability of all the child nodes given the parent node.

The notion of a history-based generative model for statistical parsing was first proposed by researchers at IBM as a complement to hand-crafted grammars (Black, Garside, and Leech 1993). The kind of model exemplified above is sometimes referred to as *head-driven*, given the central role played by syntactic heads, and this type of model is found in many state-of-the-art systems

for statistical parsing using phrase structure representations (Collins 1997; Collins 1999; Charniak 2000), dependency representations (Collins 1996; ?), and representations from specific theoretical frameworks such as TAG (Chiang 2000), HPSG (Toutanova, Manning, Shieber, Flickinger, and Oepen 2002), and CCG (Hockenmaier 2003). In addition to top-down head-driven models, there are also history-based models that use derivation steps corresponding to a particular parsing algorithm, such as left-corner derivations (Henderson 2004) or transition-based dependency parsing (Titov and Henderson 2007).

Summing up, in a generative, history-based parsing model, the generative component $\text{GEN}(x)$ is defined by a (stochastic) system of derivations that is not necessarily constrained by a formal grammar. As a consequence, the number of candidate analyses in $\text{GEN}(x)$ is normally much larger than for a simple treebank grammar. The evaluative component $\text{EVAL}(y)$ is a multiplicative model of the joint probability $P(x, y)$, factored into the conditional probability $P(d_i | \Phi(d_1, \dots, d_{i-1}))$ of each derivation step d_i given relevant parts of the derivation history.

The learning problem for these models therefore consists in estimating the conditional probabilities of different derivation steps, a problem that can be solved using relative frequency estimation as described earlier for PCFGs. However, because of the added complexity of the models, the data will be much more sparse and hence the need for smoothing more pressing. The standard approach for dealing with this problem is to back off to more general events, for example, from bilexical to monolexical probabilities, and from lexical items to parts of speech. An alternative to relative frequency estimation is to use a discriminative training technique, where parameters are set to maximize the conditional probability of the output trees given the input strings, instead of the joint probability of trees and strings. Discriminative training of generative models has sometimes been shown to improve parsing accuracy (Johnson 2001; Henderson 2004).

The inference problem, although conceptually the same, is generally harder for a history-based model than for a simple treebank PCFG, which means that there is often a trade-off between accuracy in disambiguation and efficiency in processing. For example, whereas computing the most probable analysis can be done in $O(n^3)$ time with an unlexicalized PCFG, a straightforward application of the same techniques to a fully lexicalized model takes $O(n^5)$ time, although certain optimizations are possible (cf. the Syntactic Parsing Chapter). Moreover, the greatly increased number of candidate analyses due to the lack of hard grammar constraints means that, even if parsing does not become intractable in principle, the time required for an exhaustive search of the analysis space is no longer practical. In practice, most systems of this kind only apply the full probabilistic model to a subset of all possible analyses, resulting from a first pass based on an efficient approximation of the full model. This first pass is normally implemented as some kind of chart parsing with beam search, using an estimate of the final probability to prune the search space (Caraballo and Charniak 1998).

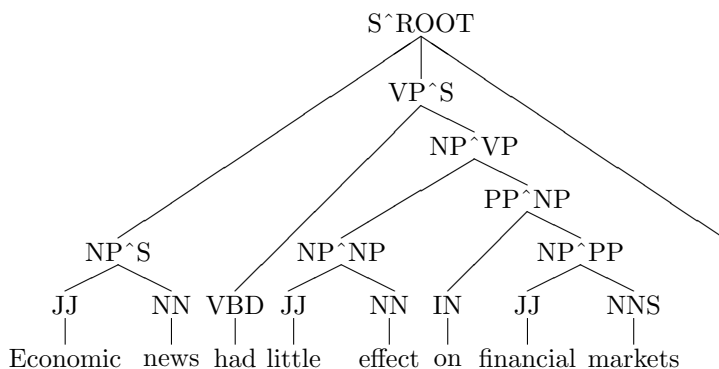


FIGURE 1.5: Constituent structure with parent annotation (cf. Figure 1.1).

1.4.2 PCFG Transformations

Although history-based models were originally conceived as an alternative (or complement) to standard PCFGs, it has later been shown that many of the dependencies captured in history-based models can in fact be modeled in a plain PCFG, provide that suitable transformations are applied to the basic treebank grammar (Johnson 1998; Klein and Manning 2003). For example, if a nonterminal node NP with parent S is instead labeled NP^S, then the dependence on structural context noted earlier in connection with pronominal NPs can be modeled in a standard PCFG, since the grammar will have different parameters for the two rules NP^S → PRP and NP^VP → PRP. This simple technique, known as *parent annotation*, has been shown to dramatically improve the parsing accuracy achieved with a simple treebank grammar (Johnson 1998). It is illustrated in Figure 1.5, which shows a version of the tree in Figure 1.1, where all the nonterminal nodes except preterminals have been reannotated in this way.

Parent annotation is an example of the technique known as *state splitting*, which consists in splitting the coarse linguistic categories that are often found in treebank annotation into more fine-grained categories that are better suited for disambiguation. An extreme example of state splitting is the use of lexicalized categories of the form $A(a)$ that we saw earlier in connection with head-driven history-based models, where nonterminal categories are split into one distinct subcategory for each possible lexical head. Exhaustive lexicalization and the modeling of bilocal relations, that is, relations holding between two lexical heads, were initially thought to be an important explanation for the success of these models, but more recent research has called this into question by showing that these relations are rarely used by the parser and account for a very small part of the increase in accuracy compared to simple treebank grammars (Gildea 2001; Bikel 2004).

These results suggest that what is important is that coarse categories are split into finer and more discriminative subcategories, which may sometimes correspond to lexicalized categories but may also be considerably more coarse-grained. Thus, in an often cited study, Klein and Manning (2003) showed that a combination of carefully defined state splits and other grammar transformations could give almost the same level of parsing accuracy as the best lexicalized parsers at the time. More recently, models have been proposed where nonterminal categories are augmented with latent variables so that state splits can be learned automatically using unsupervised learning techniques such as EM (Matsuzaki, Miyao, and Tsujii 2005; Prescher 2005; Petrov, Barrett, Thibaux, and Klein 2006; Dreyer and Eisner 2006; Liang, Petrov, Jordan, and Klein 2007; Petrov and Klein 2007). For phrase structure parsing, these latent variable models have now achieved the same level of performance as fully lexicalized generative models (Petrov, Barrett, Thibaux, and Klein 2006; Petrov and Klein 2007). An attempt to apply the same technique to dependency parsing, using PCFG transformations, did not achieve the same success (Musillo and Merlo 2008), which suggests that bilocal relations are more important in syntactic representations that lack nonterminal categories other than parts of speech.

One final type of transformation that is widely used in PCFG parsing is *markovization*, which transforms an n -ary grammar rule into a set of unary and binary rules, where each child node in the original rule is introduced in a separate rule, and where augmented nonterminals are used to encode elements of the derivation history. For example, the rule $VP \rightarrow VB\ NP\ PP$ could be transformed into:

$$\begin{aligned} VP &\rightarrow \langle VP:VB \dots PP \rangle & (1.10) \\ \langle VP:VB \dots PP \rangle &\rightarrow \langle VP:VB \dots NP \rangle PP \\ \langle VP:VB \dots NP \rangle &\rightarrow \langle VP:VB \rangle NP \\ \langle VP:VB \rangle &\rightarrow VB \end{aligned}$$

The first unary rule expands VP into a new symbol $\langle VP:VB \dots PP \rangle$, signifying a VP with head child VB and rightmost child PP . The second binary rule generates the PP child next to a child labeled $\langle VP:VB \dots NP \rangle$, representing a VP with head child VB and rightmost child NP . The third rule generates the NP child, and the fourth rule finally generates the head child VB . In this way, we can use a standard PCFG to model a head-driven stochastic process.

Grammar transformations such as markovization and state splitting make it possible to capture the essence of history-based models without formally going beyond the PCFG model. This is a distinct advantage, because it means that all the theoretical results and methods developed for PCFGs can be taken over directly. Once we have fixed the set of nonterminals and rules in the grammar, whether by ingenious hand-crafting or by learning over latent variables, we can use standard methods for learning and inference, as described earlier in Section 1.3.3. However, it is important to remember that transformations can have quite dramatic effects on the number of nonterminals and rules in the

grammar, and this in turn has a negative effect on parsing efficiency. Thus, even though exact inference for a PCFG is feasible in $O(n^3 \cdot |R|)$ (where $|R|$ is the number of grammar rules), heavy pruning is often necessary to achieve reasonable efficiency in practice.

1.4.3 Data-Oriented Parsing

An alternative approach to increasing the structural sensitivity of generative models for statistical parsing is the framework known as Data-Oriented Parsing (DOP) (Scha 1990; Bod 1995; Bod 1998; Bod 2003; Bod, Scha, and Sima'an 2003). The basic idea in the DOP model is that new sentences are parsed by combining fragments of the analyses of previously seen sentences, typically represented by a training sample from a treebank.³ This idea can be (and has been) implemented in many ways, but the standard version of DOP can be described as follows (Bod 1998):

- The set $\text{GEN}(x)$ of candidate analyses for a given sentence x is defined by a *tree substitution grammar* over all subtrees of parse trees in the training sample.
- The score $\text{EVAL}(y)$ of a given analysis $y \in \mathcal{Y}$ is the joint probability $P(x, y)$, which is equal to the sum of probabilities of all derivations of y in the tree substitution grammar.

A tree substitution grammar is a quadruple (Σ, N, S, T) , where Σ , N and S are just like in a CFG, and T is a set of *elementary trees* having root and internal nodes labeled by elements of N and leaves labeled by elements of $\Sigma \cup N$. Two elementary trees α and β can be combined by the substitution operation $\alpha \circ \beta$ to produce a unified tree only if the root of β has the same label as the leftmost nonterminal node in α , in which case $\alpha \circ \beta$ is the tree obtained by replacing the leftmost nonterminal node in α by β . The tree language $T(G)$ generated by a tree substitution grammar G is the set of all trees with root label S that can be derived using substitution of elementary trees. In this way, an ordinary CFG can be thought of as a tree substitution grammar where all elementary trees have depth 1.

This kind of model has been applied to a variety of different linguistic representations, including lexical-functional representations (Bod and Kaplan 1998) and compositional semantic representations (Bonnema, Bod, and Scha 1997), but most of the work has been concerned with syntactic parsing using phrase structure trees. Characteristic of all these models is the fact that one and the same analysis typically has several distinct derivations in the tree substitution grammar. This means that the probability $P(x, y)$ has to be computed as a sum over all derivations d that derives y ($d \Rightarrow y$), and

³There are also unsupervised versions of DOP, but we will leave them until Section 1.6.

the probability of a derivation d is normally taken to be the product of the probabilities of all subtrees t used in d ($t \in d$):

$$P(x, y) = \sum_{d \Rightarrow y} \prod_{t \in d} P(t) \quad (1.11)$$

This assumes that the subtrees of a derivation are independent of each other, just as the local trees defined by production rules are independent in a PCFG derivation. The difference is that subtrees in a DOP derivation can be of arbitrary size and can therefore capture dependencies that are outside the scope of a PCFG. A consequence of the sum-of-products model is also that the most probable analysis may not be the analysis with the most probable derivation, a property that appears to be beneficial with respect to parsing accuracy but that unfortunately makes exact inference intractable.

The learning problem for the DOP model consists in estimating the probabilities of subtrees, where the most common approach has been to use relative frequency estimation, that is, setting the probability of a subtree equal to the number of times that it is seen in the training sample divided by the number of subtrees with the same root label (Bod 1995; Bod 1998). Although this method seems to work fine in practice, it has been shown to produce a biased and inconsistent estimator (Johnson 2002), and other methods have therefore been proposed in its place (Bonnema, Buying, and Scha 2000; Bonnema and Scha 2003; Zollmann and Sima'an 2005).

As already noted, inference is a hard problem in the DOP model. Whereas computing the most probable *derivation* can be done in polynomial time, computing the most probable *analysis* (which requires summing over all derivations) is NP complete (Sima'an 1996a; Sima'an 1999). Research on efficient parsing within the DOP framework has therefore focused on finding efficient approximations that preserve the advantage gained in disambiguation by considering several distinct derivations of the same analysis. While early work focused on a kind of randomized search strategy called Monte Carlo disambiguation (Bod 1995; Bod 1998), the dominant strategy has now become the use of different kinds of PCFG reductions (Sima'an 1996b; Goodman 1996; Bod 2001; Bod 2003). This again underlines the centrality of the PCFG model for generative approaches to statistical parsing.

1.5 Discriminative Models

The statistical parsing models considered in the previous section are all generative in the sense that they model the joint probability $P(x, y)$ of the input x and output y (which in many cases is equivalent to $P(y)$). Because of this, there is often a tight integration between the system of derivations defining

$\text{GEN}(x)$ and the parameters of the scoring function $\text{EVAL}(y)$. Generative models have many advantages, such as the possibility of deriving the related probabilities $P(y|x)$ and $P(x)$ through conditionalization and marginalization, which makes it possible to use the same model for both parsing and language modeling. Another attractive property is the fact that the learning problem for these models often has a clean analytical solution, such as the relative frequency estimation for PCFGs, which makes learning both simple and efficient.

The main drawback with generative models is that they force us to make rigid independence assumptions, thereby severely restricting the range of dependencies that can be taken into account for disambiguation. As we have seen in the previous section, the search for more adequate independence assumptions has been an important driving force in research on statistical parsing, but we have also seen that more complex models inevitably makes parsing computationally harder and that we must therefore often resort to approximate algorithms. Finally, it has been pointed out that the usual approach to training a generative statistical parser maximizes a quantity – usually the joint probability of inputs and outputs in the training set – that is only indirectly related to the goal of parsing, that is, to maximize the accuracy of the parser on unseen sentences.

A discriminative model only makes use of the conditional probability $P(y|x)$ of a candidate analysis y given the input sentence x . Although this means that it is no longer possible to derive the joint probability $P(x, y)$, it has the distinct advantage that we no longer need to assume independence between features that are relevant for disambiguation and can incorporate more global features of syntactic representations. It also means that the evaluative component $\text{EVAL}(y)$ of the parsing model is not directly tied to any particular generative component $\text{GEN}(x)$, as long as we have some way of generating a set of candidate analyses. Finally, it means that we can train the model to maximize the probability of the output given the input or even to minimize a loss function in mapping inputs to outputs. On the downside, it must be said that these training regimes normally require the use of numerical optimization techniques, which can be computationally very intensive.

In discussing discriminative parsing models, we will make a distinction between *local* and *global* models. Local discriminative models try to maximize the probability of local decisions in the derivation of an analysis y , given the input x , hoping to find a globally optimal solution by making a sequence of locally optimal decisions. Global discriminative models instead try to maximize the probability of a complete analysis y , given the input x . As we shall see, local discriminative models can often be regarded as discriminative versions of generative models, with local decisions given by independence assumptions, while global discriminative models more fully exploit the potential of having features of arbitrary complexity.

1.5.1 Local Discriminative Models

Local discriminative models generally take the form of conditional history-based models, where the derivation of a candidate analysis y is modeled as a sequence of decisions with each decision conditioned on relevant parts of the derivation history. However, unlike their generative counterparts described in section 1.4.1, they also include the input sentence x as a conditioning variable:

$$P(y|x) = \prod_{i=1}^m P(d_i | \Phi(d_1, \dots, d_{i-1}, x)) \quad (1.12)$$

This makes it possible to condition decisions on arbitrary properties of the input, for example by using a lookahead such that the next k tokens of the input sentence can influence the probability of a given decision. Therefore, conditional history-based models have often been used to construct incremental and near-deterministic parsers that parse a sentence in a single left-to-right pass over the input, using beam search or some other pruning strategy to efficiently compute an approximation of the most probable analysis y given the input sentence x .

In this kind of setup, it is not strictly necessary to estimate the conditional probabilities exactly, as long as the model provides a ranking of the alternatives in terms of decreasing probability. Sometimes a distinction is therefore made between *conditional models*, where probabilities are modeled explicitly, and *discriminative models* proper, that rank alternatives without computing their probability (Jebara 2004). A special case of (purely) discriminative models are those used by deterministic parsers, such as the transition-based dependency parsers discussed below, where only the mode of the conditional distribution (i.e., the single most probable alternative) needs to be computed for each decision.

Conditional history-based models were first proposed in phrase structure parsing, as a way of introducing more structural context for disambiguation compared to standard grammar rules (Briscoe and Carroll 1993; Carroll and Briscoe 1996; Jelinek, Lafferty, Magerman, Mercer, Ratnaparkhi, and Roukos 1994; Magerman 1995). Today it is generally considered that, although parsers based on such models can be implemented very efficiently to run in linear time (Ratnaparkhi 1997; Ratnaparkhi 1999; Sagae and Lavie 2005; Sagae and Lavie 2006a), their accuracy lags a bit behind the best-performing generative models and global discriminative models. Interestingly, the same does not seem to hold for dependency parsing, where local discriminative models are used in some of the best performing systems known as *transition-based* dependency parsers (Yamada and Matsumoto 2003; Nivre, Hall, and Nilsson 2004; Isozaki, Kazawa, and Hirao 2004; Nivre 2006b; Atardi 2006; Nivre *pear*). Let us briefly consider the architecture of such a system.

We begin by noting that a dependency structure of the kind depicted in Figure 1.2 can be defined as a labeled, directed tree $y = (V, A)$, where the set

V of nodes is simply the set of tokens in the input sentence (indexed by their linear position in the string); A is a set of labeled, directed arcs (w_i, l, w_j) , where w_i, w_j are nodes and l is a dependency label (such as SBJ, OBJ); and every node except the root node has exactly one incoming arc.

A *transition system* for dependency parsing consists of a set C of configurations, representing partial analyses of sentences, and a set D of transitions from configurations to new configurations. For every sentence $x = w_1, \dots, w_n$, there is a unique initial configuration $c_i(x) \in C$ and a set $C_t(x) \subseteq C$ of terminal configurations, each representing a complete analysis y of x .

For example, if we let a configuration be a triple $c = (\sigma, \beta, A)$, where σ is a stack of nodes/tokens, β is a buffer of remaining input nodes/tokens, and A is a set of labeled dependency arcs, then we can define a transition system for dependency parsing as follows:

- The initial configuration $c_i(x) = ([], [w_1, \dots, w_n], \emptyset)$
- The set of terminal configurations $C_t(x) = \{c \in C \mid c = ([], [w_i], A)\}$
- The set D of transitions include:
 1. Shift: $(\sigma, [w_i|\beta], A) \Rightarrow ([\sigma|w_i], \beta, A)$
 2. Right-Arc(l): $([\sigma|w_i, w_j], \beta, A) \Rightarrow ([\sigma|w_i], \beta, A \cup \{(w_i, l, w_j)\})$
 3. Left-Arc(l): $([\sigma|w_i, w_j], \beta, A) \Rightarrow ([\sigma|w_j], \beta, A \cup \{(w_j, l, w_i)\})$

The initial configuration has an empty stack, an empty arc set, and all the input tokens in the buffer. A terminal configuration has a single token on the stack and an empty buffer. The Shift transition moves the next token in the buffer onto the stack, while the Right-Arc(l) and Left-Arc(l) transitions add a dependency arc between the two top tokens on the stack and replace them by the head token of that arc. It is easy to show that, for any sentence $x = w_1, \dots, w_n$ with a projective dependency tree y ,⁴ there is a transition sequence that builds y in exactly $2n - 1$ steps starting from $c_i(x)$. Over the years, a number of different transition systems have been proposed for dependency parsing, some of which are restricted to projective dependency trees (Kudo and Matsumoto 2002; Yamada and Matsumoto 2003; Nivre 2003), while others can also derive non-projective structures (Attardi 2006; Nivre 2006a; Nivre 2007).

Given a scoring function $S(\Phi(c), d)$, which scores possible transitions d out of a configuration c , represented by a high-dimensional feature vector $\Phi(c)$, and given a way of combining the scores of individual transitions into scores for complete sequences, parsing can be performed as search for the highest-scoring transition sequence. Different search strategies are possible, but most transition-based dependency parsers implement some form of beam search,

⁴A dependency tree is projective iff every subtree has a contiguous yield.

with a fixed constant beam width k , which means that parsing can be performed in $O(n)$ time for transition systems where the length of a transition sequence is linear in the length of the sentence. In fact, many systems set k to 1, which means that parsing is completely deterministic given the scoring function. If the scoring function $S(\Phi(c), d)$ is designed to estimate (or maximize) the conditional probability of a transition d given the configuration c , then this is a local, discriminative model. It is discriminative because the configuration c encodes properties both of the input sentence and of the transition history; and it is local because each transition d is scored in isolation.

Summing up, in statistical parsers based on local, discriminative models, the generative component $\text{GEN}(x)$ is typically defined by a derivational process, such as a transition system or a bottom-up parsing algorithm, while the evaluative component $\text{EVAL}(y)$ is essentially a model for scoring local decisions, conditioned on the input and parts of the derivation history, together with a way of combining local scores into global scores.

The learning problem for these models is to learn a scoring function for local decisions, conditioned on the input and derivation history, a problem that can be solved using many different techniques. Early history-based models for phrase structure parsing used decision tree learning (Jelinek, Lafferty, Magerman, Mercer, Ratnaparkhi, and Roukos 1994; Magerman 1995), but more recently log-linear models have been the method of choice (Ratnaparkhi 1997; Ratnaparkhi 1999; Sagae and Lavie 2005; Sagae and Lavie 2006a). The latter method has the advantage that it gives a proper, conditional probability model, which facilitates the combination of local scores into global scores. In transition-based dependency parsing, purely discriminative approaches such as support vector machines (Kudo and Matsumoto 2002; Yamada and Matsumoto 2003; Isozaki, Kazawa, and Hirao 2004; Nivre, Hall, Nilsson, Eryigit, and Marinov 2006), perceptron learning (Ciaramita and Attardi 2007), and memory-based learning (Nivre, Hall, and Nilsson 2004; Attardi 2006) have been more popular, although log-linear models have been used in this context as well (Cheng, Asahara, and Matsumoto 2005; Attardi 2006).

The inference problem is to compute the optimal decision sequence, given the scoring function, a problem that is usually tackled by some kind of approximate search, such as beam search (with greedy, deterministic search as a special case). This guarantees that inference can be performed efficiently even with exponentially many derivations and a model structure that is often unsuited for dynamic programming. As already noted, parsers based on local, discriminative models can be made to run very efficiently, often in linear time, either as a theoretical worst-case (Nivre 2003; Sagae and Lavie 2005) or as an empirical average-case (Ratnaparkhi 1997; Ratnaparkhi 1999).

1.5.2 Global Discriminative Models

In a local discriminative model, the score of an analysis y , given the sentence x , factors into the scores of different decisions in the derivation of y . In a

global discriminative model, by contrast, no such factorization is assumed, and component scores can all be defined on the entire analysis y . This has the advantage that the model may incorporate features that capture global properties of the analysis, without being restricted to a particular history-based derivation of the analysis (whether generative or discriminative).

In a global discriminative model, a scoring function $S(x, y)$ is typically defined as the inner product of a feature vector $\mathbf{f}(x, y) = \langle f_1(x, y), \dots, f_k(x, y) \rangle$ and a weight vector $\mathbf{w} = \langle w_1, \dots, w_m \rangle$:

$$S(x, y) = \mathbf{f}(x, y) \cdot \mathbf{w} = \sum_{i=1}^k w_i \cdot f_i(x, y) \quad (1.13)$$

where each $f_i(x, y)$ is a (numerical) feature of x and y , and each w_i is a real-valued weight quantifying the tendency of feature $f_i(x, y)$ to co-occur with optimal analyses. A positive weight indicates a positive correlation, a negative weight indicates a negative correlation, and by summing up all feature-weight products we obtain a global estimate of the optimality of the analysis y for sentence x .

The main strength of this kind of model is that there are no restrictions on the kind of features that may be used, except that they must be encoded as numerical features. For example, it is perfectly straightforward to define features indicating the presence or absence of a particular substructure, such as the tree of depth 1 corresponding to a PCFG rule. In fact, we can represent the entire scoring function of the standard PCFG model by having one feature $f_i(x, y)$ for each grammar rule r_i , whose value is the number of times r_i is used in the derivation of y , and setting w_i to the log of the rule probability for r_i . The global score will then be equivalent to the log of the probability $P(x, y)$ as defined by the corresponding PCFG, in virtue of the following equivalence:

$$\log \left[\prod_{i=1}^{|R|} D(r_i)^{\text{COUNT}(i, y)} \right] = \sum_{i=1}^{|R|} \log D(r_i) \cdot \text{COUNT}(i, y) \quad (1.14)$$

However, the main advantage of these models lies in features that go beyond the capacity of local models and capture more global properties of syntactic structures, for example, features that indicate conjunct parallelism in coordinate structures, features that encode differences in length between conjuncts, features that capture the degree of right branching in a parse tree, or features that signal the presence of “heavy” constituents of different types (Charniak and Johnson 2005). It is also possible to use features that encode the scores assigned to a particular analysis by other parsers, which means that the model can also be used as a framework for parser combination.

The learning problem for a global discriminative model is to estimate the weight vector \mathbf{w} . This can be solved using setting the weights to maximize the conditional likelihood of the preferred analyses in the training data according

to the following model:

$$P(y|x) = \frac{\exp[\mathbf{f}(x, y) \cdot \mathbf{w}]}{\sum_{y' \in \text{GEN}(x)} \exp[\mathbf{f}(x, y') \cdot \mathbf{w}]} \quad (1.15)$$

The exponentiated score of analysis y for sentence x is normalized to a conditional probability by dividing it with the sum of exponentiated scores of all alternative analyses $y' \in \text{GEN}(x)$. This kind of model is usually called a log-linear model, or an exponential model. The problem of finding the optimal weights has no closed form solution, but there are a variety of numerical optimization techniques that can be used, including iterative scaling and conjugate gradient techniques, making log-linear models one of the most popular choices for global discriminative models (Johnson, Geman, Canon, Chi, and Riezler 1999; Riezler, King, Kaplan, Crouch, Maxwell III, and Johnson 2002; Toutanova, Manning, Shieber, Flickinger, and Oepen 2002; Miyao, Ninomiya, and Tsujii 2003; Clark and Curran 2004).

An alternative approach is to use a purely discriminative learning method, which does not estimate a conditional probability distribution but simply tries to separate the preferred analyses from alternative analyses, setting the weights so that the following criterion is upheld for every sentence x with preferred analysis y in the training set:

$$y = \arg \max_{y' \in \text{GEN}(x)} \mathbf{f}(x, y') \cdot \mathbf{w} \quad (1.16)$$

In case the set of constraints is not satisfiable, techniques such as slack variables can be used to allow some constraints to be violated with a penalty. Methods in this family include the perceptron algorithm and max-margin methods such as support vector machines, which are also widely used in the literature (Collins 2000; Collins and Duffy 2002; Taskar, Klein, Collins, Koller, and Manning 2004; Collins and Koo 2005; McDonald, Crammer, and Pereira 2005). Common to all of these methods, whether conditional or discriminative, is the need to repeatedly reparse the training corpus, which makes learning of global discriminative models computationally intensive.

The use of truly global features is an advantage from the point of view of parsing accuracy but has the drawback of making inference intractable in the general case. Since there is no restriction on the scope that features may take, it is not possible to use standard dynamic programming techniques to compute the optimal analysis. This is relevant not only at parsing time but also during learning, given the need to repeatedly reparse the training corpus during optimization.

The most common way of dealing with this problem is to use a different model to define $\text{GEN}(x)$ and to use the inference method for this base model to derive what is typically a restricted subset of all candidate analyses. This approach is especially natural in grammar-driven systems, where the base parser is used to derive the set of candidates that are compatible with the constraints of the grammar, and the global discriminative model is applied only to

this subset. This methodology underlies many of the best performing broad-coverage parsers for theoretical frameworks such as LFG (Johnson, Geman, Canon, Chi, and Riezler 1999; Riezler, King, Kaplan, Crouch, Maxwell III, and Johnson 2002), HPSG (Toutanova, Manning, Shieber, Flickinger, and Oepen 2002; Miyao, Ninomiya, and Tsujii 2003) and CCG (Clark and Curran 2004; Curran and Clark 2004), some of which are based on hand-crafted grammars while others use theory-specific treebank grammars.

The two-level model is also commonly used in data-driven systems, where the base parser responsible for the generative component $\text{GEN}(x)$ is typically a parser using a generative model. These parsers are known as *reranking* parsers, since the global discriminative model is used to rerank the k top candidates already ranked by the generative base parser. Applying a discriminative reranker on top of a generative base parser usually leads to a significant improvement in parsing accuracy (Collins 2000; Collins and Duffy 2002; Collins and Koo 2005; Charniak and Johnson 2005). However, it is worth noting that the single most important feature in the global discriminative model is normally the log probability assigned to an analysis by the generative base parser.

A potential problem with the standard reranking approach to discriminative parsing is that $\text{GEN}(x)$ is usually restricted to a small subset of all possible analyses, which means that the truly optimal analysis may not even be included in the set of analyses that are considered by the discriminative model. That this is a real problem was shown in the study of Collins (2000), where 41% of the correct analyses were not included in the set of 30 best parses considered by the reranker. In order to overcome this problem, discriminative models with global inference have been proposed, either using dynamic programming and restricting the scope of features (Taskar, Klein, Collins, Koller, and Manning 2004) or using approximate search (Turian and Melamed 2006), but efficiency remains a problem for these methods, which do not seem to scale up to sentences of arbitrary length. A recent alternative is *forest reranking* (Huang 2008), a method that reranks a packed forest of trees, instead of complete trees, and uses approximate inference to make training tractable.

The efficiency problems associated with inference for global discriminative models are most severe for phrase structure representations and other more expressive formalisms. Dependency representations, by contrast, are more tractable in this respect, and one of the most successful approaches to dependency parsing in recent years, known as *spanning tree* parsing (or *graph-based* parsing), is based on exact inference with global, discriminative models.

The starting point for spanning tree parsing is the observation that the set $\text{GEN}(x)$ of all dependency trees for a sentence x (given some set of dependency labels) can be compactly represented as a dense graph $G = (V, A)$, where V is the set of nodes corresponding to tokens of x , and A contains all possible labeled directed arcs (w_i, l, w_j) connecting nodes in V . Given a model for scoring dependency trees, the inference problem for dependency parsing then becomes the problem of finding the highest scoring spanning tree in G

(McDonald, Pereira, Ribarov, and Hajič 2005).

With suitably factored models, the optimum spanning tree can be computed in $O(n^3)$ time for projective dependency trees using Eisner’s algorithm (Eisner 1996; Eisner 2000), and in $O(n^2)$ time for arbitrary dependency trees using the Chu-Liu-Edmonds algorithm (Chu and Liu 1965; Edmonds 1967). This makes global discriminative training perfectly feasible, and spanning tree parsing has become one of the dominant paradigms for statistical dependency parsing (McDonald, Crammer, and Pereira 2005; McDonald, Pereira, Ribarov, and Hajič 2005; McDonald and Pereira 2006; Carreras 2007). Although exact inference is only possible if features are restricted to small subgraphs (even single arcs if non-projective trees are allowed), various techniques have been developed for approximate inference with more global features (McDonald and Pereira 2006; Riedel, Çakıcı, and Meza-Ruiz 2006; Nakagawa 2007). Moreover, using a generalization of the Chu-Liu-Edmonds algorithms to k -best parsing, it is possible to add a discriminative reranker on top of the discriminative spanning tree parser (Hall 2007).

To conclude, the common denominator of the models discussed in this section is an evaluative component where the score $\text{EVAL}(y)$ is defined by a linear combination of weighted features that are not restricted by a particular derivation process, and where weights are learned using discriminative techniques such as conditional likelihood estimation or perceptron learning. Exact inference is intractable in general, which is why the set $\text{GEN}(x)$ of candidates is often restricted to a small set generated by a grammar-driven or generative statistical parser, a set that can be searched exhaustively. Exact inference has so far been practically useful mainly in the context of graph-based dependency parsing.

1.6 Beyond Supervised Parsing

All the methods for statistical parsing discussed so far in this chapter rely on supervised learning in some form. That is, they need to have access to sentences labeled with their preferred analyses in order to estimate model parameters. As noted in the introduction, this is a serious limitation, given that there are few languages in the world for which there exist any syntactically annotated data, not to mention the wide range of domains and text types for which no labeled data are available even in well-resourced languages such as English. Consequently, the development of methods that can learn from unlabeled data, either alone or in combination with labeled data, should be of primary importance, even though it has so far played a rather marginal role in the statistical parsing community. In this final section, we will briefly review some of the existing work in this area.

1.6.1 Weakly Supervised Parsing

Weakly supervised (or semi-supervised) learning refers to techniques that use labeled data as in supervised learning but complements this with learning from unlabeled data, usually in much larger quantities than the labeled data, hence reducing the need for manual annotation to produce labeled data. The most common approach is to use the labeled data to train one or more systems that can then be used to label new data, and to retrain the systems on a combination of the original labeled data and the new, automatically labeled data. One of the key issues in the design of such a method is how to decide which automatically labeled data instances to include in the new training set.

In *co-training* (Blum and Mitchell 1998), two or more systems with complementary views of the data are used, so that each data instance is described using two different feature sets that provide different, complementary information about the instance. Ideally, the two views should be conditionally independent and each view sufficient by itself. The two systems are first trained on the labeled data and used to analyze the unlabeled data. The most confident predictions of each system on the unlabeled data are then used to iteratively construct additional labeled training data for the other system. Co-training has been applied to syntactic parsing but the results so far are rather mixed (Sarkar 2001; Steedman, Hwa, Osborne, and Sarkar 2003). One potential use of co-training is in domain adaptation, where systems have been trained on labeled out-of-domain data and need to be tuned using unlabeled in-domain data. In this setup, a simple variation on co-training has proven effective, where an automatically labeled instance is added to the new training set only if both systems agree on its analysis (Sagae and Tsujii 2007).

In *self-training*, one and the same system is used to label its own training data. According to the received wisdom, this scheme should be less effective than co-training, given that it does not provide two independent views of the data, and early studies of self-training for statistical parsing seemed to confirm this (Steedman, Hwa, Osborne, and Sarkar 2003; Charniak 1997). More recently, however, self-training has been used successfully to improve parsing accuracy on both in-domain and out-of-domain data (McClosky, Charniak, and Johnson 2006a; McClosky, Charniak, and Johnson 2006b; McClosky, Charniak, and Johnson 2008). It seems that more research is needed to understand the conditions that are necessary in order for self-training to be effective (McClosky, Charniak, and Johnson 2008).

1.6.2 Unsupervised Parsing

Unsupervised parsing amounts to the induction of a statistical parsing model from raw text. Early work in this area was based on the PCFG model, trying to learn rule probabilities for a fixed-form grammar using the Inside-Outside algorithm (Baker 1979; Lari and Young 1990) but with rather limited success (Carroll and Charniak 1992; Pereira and Schabes 1992). More recent work

has instead focused on models inspired by successful approaches to supervised parsing, in particular history-based models and data-oriented parsing.

As an example, let us consider the Constituent-Context Model (CCM) (Klein and Manning 2002; Klein 2005). Let $x = w_1, \dots, w_n$ be a sentence, let y be a tree for x , and let y_{ij} be **true** if w_i, \dots, w_j is a constituent according to y and **false** otherwise. The joint probability $P(x, y)$ of a sentence x and a tree y is equivalent to $P(y)P(x|y)$, where $P(y)$ is the a priori probability of the tree (usually assumed to come from a uniform distribution), and $P(x|y)$ is modeled as follows:

$$P(x|y) = \prod_{1 \leq i < j \leq n} P(w_i, \dots, w_j | y_{ij}) P(w_{i-1}, w_{j+1} | y_{ij}) \quad (1.17)$$

The two conditional probabilities on the right-hand side of the equation are referred to as the *constituent* and the *context* probabilities, respectively, even though they are defined not only for constituents but for all spans of the sentence x . Using the EM algorithm to estimate the parameters of the constituent and context distributions resulted in the first model to beat the right-branching baseline when evaluated on the data set known as WSJ10, consisting of part-of-speech sequences for all sentences up to length 10 in the Wall Street Journal section of the Penn Treebank (Klein and Manning 2002; Klein 2005).⁵

The results can be improved further by combining CCM with the dependency-based DMV model, which is inspired by the head-driven history-based models described in Section 1.4.1 (Klein and Manning 2004; Klein 2005) and further discussed below.

Another class of models that have achieved competitive results are the various unsupervised versions of the DOP model (cf. section 1.4.3). Whereas learning in the supervised DOP model takes into account all possible subtrees of the preferred parse tree y for sentence x , learning in the unsupervised setting takes into account all possible subtrees of *all* possible parse trees of x . There are different ways to train such a model, but applying the EM algorithm to an efficient PCFG reduction (the so-called UML-DOP model) gives empirical results on a par with the combined CCM+DMV model (Bod 2007).

An alternative approach to unsupervised phrase structure parsing is the common-cover-link model (Seginer 2007), which uses a linear-time incremental parsing algorithm for a link-based representation of phrase structure and learns from very simple surface statistics. Unlike the other models discussed in this section, this model is efficient enough to learn from plain words (not part-of-speech tags) without any upper bound on sentence length. Although evaluation results are not directly comparable, the common-cover-link model appears to give competitive accuracy.

⁵The right-branching baseline assigns to every sentence a strictly right-branching, binary tree. Since parse trees for English are predominantly right-branching, this constitutes a rather demanding baseline for unsupervised parsing.

The work discussed so far has all been concerned with unsupervised phrase structure parsing, but there has also been work on inducing models for dependency parsing. Early proposals involved models that start by generating an abstract dependency tree (without words) and then populate the tree with words according to a distribution where each word is conditioned only on its head and on the direction of attachment (Yuret 1998; Paskin 2001b; Paskin 2001a). However, a problem with this kind of model is that it tends to link words that have high mutual information regardless of whether they are plausibly syntactically related.

In order to overcome the problems of the earlier models, the DMV model mentioned earlier was proposed (Klein and Manning 2004; Klein 2005). DMV is short for Dependency Model with Valence, and the model is clearly inspired by the head-driven history-based models used in supervised parsing (cf. Section 1.4.1). In this model, a dependency (sub)tree rooted at h , denoted $T(h)$, is generated as follows:

$$P(T(h)) = \prod_{d \in \{l, r\}} \left[\prod_{a \in D(h, d)} P_{\uparrow}(\neg \uparrow | h, d, ?) P_v(a | h, d) P(T(a)) \right] P_{\uparrow}(\uparrow | h, d, ?) \quad (1.18)$$

In this equation, d is a variable over the direction of the dependency – left (l) or right (r); $D(h, d)$ is the set of dependents of h in direction d ; $P_{\uparrow}(\uparrow | h, d, ?)$ is the probability of stopping the generation of dependents in direction d and $?$ is a binary variable indicating whether any dependents have been generated or not; $P_v(a | h, d)$ is the probability of generating the dependent word a , conditioned on the head h and direction d ; and $P(T(a))$ is (recursively) the probability of the subtree rooted at a .

The DMV model has not only improved results for unsupervised dependency parsing but has also been combined with the CCM model to improve results for both phrase structure parsing and dependency parsing. The original work on the DMV model used the EM algorithm for parameter estimation (Klein and Manning 2004; Klein 2005), but results have later been improved substantially through the use of alternative estimation methods such as contrastive estimation and structured annealing (Smith 2006).

1.7 Summary and Conclusions

In this chapter, we have tried to give an overview of the most prominent approaches to statistical parsing that are found in the field today, characterizing the different models in terms of their generative and evaluative components and discussing the problems of learning and inference that they give rise to. Overall, the field is dominated by supervised approaches that make use of generative or discriminative statistical models to rank the candidate analyses for

a given input sentence. In terms of empirical accuracy, discriminative models seem to have a slight edge over generative models, especially discriminative models that incorporate global features, but it is important to remember that many discriminative parsers include the output of a generative model in their feature representations. For both generative and discriminative models, we can see a clear development towards models that take more global structure into account, which improves their capacity for disambiguation but makes parsing computationally harder. In this way, there is an inevitable tradeoff between accuracy and efficiency in statistical parsing.

Parsers that learn from unlabeled data – instead of or in addition to labeled data – have so far played a marginal role in statistical parsing but are likely to become more important in the future. Developing a large-scale treebank for every new language and domain we want to parse is simply not a scalable solution, so research on methods that do not rely (only) on labeled data is a major concern for the field. Although the empirical results in terms of parsing accuracy have so far not been on a par with those for supervised approaches, results are steadily improving. Moreover, it is clear that the comparison has been biased in favor of the supervised systems, because the outputs of both systems have been compared to the kind of representations that supervised parsers learn from. One way to get around this problem is to use application-driven evaluation instead, and there are signs that in this context unsupervised approaches can already compete with supervised approaches for some applications (Bod 2007).

Finally, it is worth pointing out that this survey of statistical parsing is by no means exhaustive. We have chosen to concentrate on the types of models that have been important for driving the development of the field and that are also found in the best performing systems today, without describing any of the systems in detail. One topic that we have not touched upon at all is *system combination*, that is, techniques for improving parsing accuracy by combining several models, either at learning time or at parsing time. In fact, many of the best performing parsers available today for different types of syntactic representations do in some way involve system combination. Thus, Charniak and Johnson’s reranking parser (Charniak and Johnson 2005) includes Charniak’s generative parser (Charniak 2000) as a component, and there are many dependency parsers that combine several models either by voting (Zeman and Žabokrtský 2005; Sagae and Lavie 2006b; Hall, Nilsson, Nivre, Eryigit, Megyesi, Nilsson, and Saers 2007) or by stacking (Nivre and McDonald 2008). It is likely that system combination will remain an important technique for boosting accuracy, even if single models become increasingly more accurate by themselves in the future.

Acknowledgments

I want to thank John Carroll and Jason Eisner for valuable comments on an earlier version of this chapter. I am also grateful to Peter Ljunglöf and Mats Wirén for discussions about the organization of the two parsing chapters (the Syntactic Parsing Chapter and this one).

Bibliography

- Aduriz, I., M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz (2003). Construction of a Basque dependency treebank. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 201–204.
- Attardi, G. (2006). Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pp. 166–170.
- Baker, J. (1979). Trainable grammars for speech recognition. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.
- Bikel, D. (2004). *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph. D. thesis, University of Pennsylvania.
- Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, S. Roukos, B. Santorini, and T. Strzalkowski (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pp. 306–311.
- Black, E., R. Garside, and G. Leech (Eds.) (1993). *Statistically-Driven Computer Grammars of English: The IBM/Lancaster Approach*. Rodopi.
- Blum, A. and T. Mitchell (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Workshop on Computational Learning Theory (COLT)*, pp. 92–100.
- Bod, R. (1995). *Enriching Linguistics with Statistics: Performance Models of Natural Language*. Ph. D. thesis, University of Amsterdam.
- Bod, R. (1998). *Beyond Grammar*. CSLI Publications.
- Bod, R. (2001). What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 66–73.

- Bod, R. (2003). An efficient implementation of a new DOP model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 19–26.
- Bod, R. (2007). Is the end of supervised parsing in sight? In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 400–407.
- Bod, R. and R. Kaplan (1998). A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and the 17th International Conference on Computational Linguistics (COLING)*, pp. 145–151.
- Bod, R., R. Scha, and K. Sima'an (Eds.) (2003). *Data-Oriented Parsing*. CSLI Publications.
- Boguslavsky, I., S. Grigorieva, N. Grigoriev, L. Kreidlin, and N. Frid (2000). Dependency treebank for Russian: Concept, tools, types of information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pp. 987–991.
- Böhmová, A., J. Hajič, E. Hajičová, and B. Hladká (2003). The Prague Dependency Treebank: A three-level annotation scenario. In A. Abeillé (Ed.), *Treebanks: Building and Using Parsed Corpora*, pp. 103–127. Kluwer.
- Bonnema, R., R. Bod, and R. Scha (1997). A DOP model for semantic interpretation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 159–167.
- Bonnema, R., P. Buying, and R. Scha (2000). Parse tree probability in data oriented parsing. In *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics*, pp. 219–232.
- Bonnema, R. and R. Scha (2003). Reconsidering the probability model for DOP. In R. Bod, R. Scha, and K. Sima'an (Eds.), *Data-Oriented Parsing*, pp. 25–41. CSLI Publications.
- Booth, T. L. and R. A. Thompson (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers C-22*, 442–450.
- Bresnan, J. (2000). *Lexical-Functional Syntax*. Blackwell.
- Briscoe, E. and J. Carroll (1993). Generalised probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics 19*, 25–59.
- Buchholz, S. and E. Marsi (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pp. 149–164.

- Caraballo, S. A. and E. Charniak (1998). New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics* 24, 275–298.
- Carreras, X. (2007). Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pp. 957–961.
- Carroll, G. and E. Charniak (1992). Two experiments on learning probabilistic dependency grammars from corpora. Technical Report TR-92, Department of Computer Science, Brown University.
- Carroll, J. and E. Briscoe (1996). Apportioning development effort in a probabilistic LR parsing system through evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 92–100.
- Carroll, J., E. Briscoe, and A. Sanfilippo (1998). Parser evaluation: A survey and a new proposal. In *Proceedings of the [1st] International Conference on Language Resources and Evaluation (LREC)*, pp. 447–454.
- Carroll, J., G. Minnen, and E. Briscoe (2003). Parser evaluation using a grammatical relation annotation scheme. In A. Abeillé (Ed.), *Treebanks*, pp. 299–316. Kluwer.
- Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 1031–1036.
- Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI/IAAI*, pp. 598–603.
- Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 132–139.
- Charniak, E. and M. Johnson (2005). Coarse-to-fine n -best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 173–180.
- Cheng, Y., M. Asahara, and Y. Matsumoto (2005). Machine learning-based dependency analyzer for Chinese. In *Proceedings of International Conference on Chinese Computing (ICCC)*, pp. 66–73.
- Chi, Z. and S. Geman (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics* 24, 299–305.
- Chiang, D. (2000). Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 456–463.
- Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory IT-2*, 113–124.

- Chu, Y. J. and T. H. Liu (1965). On the shortest arborescence of a directed graph. *Science Sinica* 14, 1396–1400.
- Ciaramita, M. and G. Attardi (2007, June). Dependency parsing with second-order feature maps and annotated semantic information. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pp. 133–143.
- Clark, S. and J. R. Curran (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 104–111.
- Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 184–191.
- Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 16–23.
- Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph. D. thesis, University of Pennsylvania.
- Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 175–182.
- Collins, M. and N. Duffy (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 263–270.
- Collins, M. and T. Koo (2005). Discriminative reranking for natural language parsing. *Computational Linguistics* 31, 25–71.
- Curran, J. R. and S. Clark (2004). The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 282–288.
- Dreyer, M. and J. Eisner (2006). Better informed training of latent syntactic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 317–326.
- Džeroski, S., T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele (2006). Towards a Slovene dependency treebank. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.
- Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards* 71B, 233–240.

- Eisner, J. and G. Satta (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 457–464.
- Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 340–345.
- Eisner, J. M. (2000). Bilexical grammars and their cubic-time parsing algorithms. In H. Bunt and A. Nijholt (Eds.), *Advances in Probabilistic and Other Parsing Technologies*, pp. 29–62. Kluwer.
- Forst, M., N. Bertomeu, B. Crysmann, F. Fouvry, S. Hansen-Schirra, and V. Kordoni (2004). The TIGER dependency bank. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, pp. 31–37.
- Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 167–202.
- Goodman, J. (1996). Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 177–183.
- Goodman, J. (1999). Semiring parsing. *Computational Linguistics* 25, 573–605.
- Grishman, R., C. Macleod, and J. Sterling (1992). Evaluating parsing strategies using standardized parse files. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP)*, pp. 156–161.
- Hajič, J., O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. Prague Arabic Dependency Treebank: Development in data and tools. In *Proceedings of the NEMLAR International Conference on Arabic Language Resources and Tools*.
- Hajič, J., B. Vidova Hladká, J. Panevová, E. Hajičová, P. Sgall, and P. Pajas (2001). Prague Dependency Treebank 1.0. LDC, 2001T10.
- Hall, J., J. Nilsson, J. Nivre, G. Eryiğit, B. Megyesi, M. Nilsson, and M. Saers (2007). Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*.
- Hall, K. (2007). K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 392–399.
- Han, C.-H., N.-R. Han, E.-S. Ko, and M. Palmer (2002). Development and evaluation of a Korean treebank and its application to NLP. In

Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC), pp. 1635–1642.

- Henderson, J. (2004). Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 96–103.
- Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph. D. thesis, University of Edinburgh.
- Hockenmaier, J. and M. Steedman (2007). CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33, 355–396.
- Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 586–594.
- Huang, L. and D. Chiang (2005). Better k -best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.
- Isozaki, H., H. Kazawa, and T. Hirao (2004). A deterministic word dependency analyzer enhanced with preference learning. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pp. 275–281.
- Jebara, T. (2004). *Machine Learning: Discriminative and Generative*. Kluwer.
- Jelinek, F., J. Lafferty, D. M. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos (1994). Decision tree parsing using a hidden derivation model. In *Proceedings of the ARPA Human Language Technology Workshop*, pp. 272–277.
- Jiménez, V. M. and A. Marzal (2000). Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*.
- Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics* 24, 613–632.
- Johnson, M. (2001). Joint and conditional estimation of tagging and parsing models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 314–321.
- Johnson, M. (2002). A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 136–143.
- Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler (1999). Estimators for stochastic “unification-based” grammars. In *Proceedings of the*

37th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 535–541.

- Joshi, A. (1985). How much context-sensitivity is necessary for assigning structural descriptions: Tree adjoining grammars. In D. Dowty, L. Karttunen, and A. Zwicky (Eds.), *Natural Language Processing: Psycholinguistic, Computational and Theoretical Perspectives*, pp. 206–250. Cambridge University Press.
- Joshi, A. K. (1997). Tree-adjoining grammars. In G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages. Volume 3: Beyond Words*, pp. 69–123. Springer.
- Kaplan, R. and J. Bresnan (1982). Lexical-Functional Grammar: A formal system for grammatical representation. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations*, pp. 173–281. MIT Press.
- King, T. H., R. Crouch, S. Riezler, M. Dalrymple, and R. M. Kaplan (2003). The PARC 700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, pp. 1–8.
- Klein, D. (2005). *The Unsupervised Learning of Natural Language Structure*. Ph. D. thesis, Stanford University.
- Klein, D. and C. D. Manning (2002). Conditional structure versus conditional estimation in NLP models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9–16.
- Klein, D. and C. D. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 423–430.
- Klein, D. and C. D. Manning (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 479–486.
- Kromann, M. T. (2003). The Danish Dependency Treebank and the DTAG treebank tool. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 217–220.
- Kudo, T. and Y. Matsumoto (2002). Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL)*, pp. 63–69.
- Lari, K. and S. S. Young (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4, 35–56.
- Liang, P., S. Petrov, M. Jordan, and D. Klein (2007). The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and*

- Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 688–697.
- Lin, D. (1995). A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, pp. 1420–1425.
- Lin, D. (1998). A dependency-based method for evaluating broad-coverage parsers. *Journal of Natural Language Engineering* 4, 97–114.
- Maamouri, M. and A. Bies (2004). Developing an Arabic treebank: Methods, guidelines, procedures, and tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-Based Languages*, pp. 2–9.
- Magerman, D. M. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 276–283.
- Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19, 313–330.
- Marcus, M. P., B. Santorini, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger (1994). The Penn Treebank: Annotating predicate-argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*, pp. 114–119.
- Matsuzaki, T., Y. Miyao, and J. Tsujii (2005). Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 75–82.
- McClosky, D., E. Charniak, and M. Johnson (2006a). Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 152–159.
- McClosky, D., E. Charniak, and M. Johnson (2006b). Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pp. 337–344.
- McClosky, D., E. Charniak, and M. Johnson (2008). When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pp. 561–568.
- McDonald, R., K. Crammer, and F. Pereira (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 91–98.
- McDonald, R. and F. Pereira (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 81–88.

- McDonald, R., F. Pereira, K. Ribarov, and J. Hajič (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pp. 523–530.
- Mel'čuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.
- Miyao, Y., T. Ninomiya, and J. Tsujii (2003). Probabilistic modeling of argument structures including non-local dependencies. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pp. 285–291.
- Moreno, A., S. López, F. Sánchez, and R. Grishman (2003). Developing a Spanish treebank. In A. Abeillé (Ed.), *Treebanks: Building and Using Parsed Corpora*, pp. 149–163. Kluwer.
- Musillo, G. and P. Merlo (2008). Unlexicalised hidden variable models of split dependency grammars. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 213–216.
- Nakagawa, T. (2007). Multilingual dependency parsing using global features. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pp. 952–956.
- Ney, H. (1991). Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing* 39, 336–340.
- Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 149–160.
- Nivre, J. (2006a). Constraints on non-projective dependency graphs. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 73–80.
- Nivre, J. (2006b). *Inductive Dependency Parsing*. Springer.
- Nivre, J. (2007). Incremental non-projective dependency parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pp. 396–403.
- Nivre, J. (to appear). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*.
- Nivre, J., J. Hall, and J. Nilsson (2004). Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, pp. 49–56.
- Nivre, J., J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov (2006). Labeled pseudo-projective dependency parsing with support vector machines. In

- Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pp. 221–225.
- Nivre, J. and R. McDonald (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Oflazer, K., B. Say, D. Z. Hakkani-Tür, and G. Tür (2003). Building a Turkish treebank. In A. Abeillé (Ed.), *Treebanks: Building and Using Parsed Corpora*, pp. 261–277. Kluwer.
- Paskin, M. A. (2001a). Cubic-time parsing and learning algorithms for grammatical bigram models. Technical Report UCB/CSD-01-1148, Computer Science Division, University of California Berkeley.
- Paskin, M. A. (2001b). Grammatical bigrams. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–97.
- Pereira, F. C. and Y. Schabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 128–135.
- Petrov, S., L. Barrett, R. Thibaux, and D. Klein (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pp. 433–440.
- Petrov, S. and D. Klein (2007). Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pp. 404–411.
- Pollard, C. and I. A. Sag (1987). *Information-Based Syntax and Semantics*. CSLI Publications.
- Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. CSLI Publications.
- Prescher, D. (2005). Head-driven PCFGs with latent-head statistics. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pp. 115–124.
- Prokopidis, P., E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis (2005). Theoretical and practical issues in the construction of a Greek dependency treebank. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 149–160.
- Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–10.
- Ratnaparkhi, A. (1999). Learning to parse natural language with maximum entropy models. *Machine Learning* 34, 151–175.

- Riedel, S., R. Çakıcı, and I. Meza-Ruiz (2006). Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pp. 226–230.
- Riezler, S., M. H. King, R. M. Kaplan, R. Crouch, J. T. Maxwell III, and M. Johnson (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 271–278.
- Sagae, K. and A. Lavie (2005). A classifier-based parser with linear runtime complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pp. 125–132.
- Sagae, K. and A. Lavie (2006a). A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 691–698.
- Sagae, K. and A. Lavie (2006b). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 129–132.
- Sagae, K. and J. Tsujii (2007). Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pp. 1044–1050.
- Sánchez, J. A. and J. M. Benedí (1997). Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 1052–1055.
- Sarkar, A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 175–182.
- Scha, R. (1990). Taaltheorie en taaltechnologie; competence en performance [language theory and language technology; competence and performance]. In R. de Kort and G. L. J. Leerdam (Eds.), *Computer-toepassingen in de Neerlandistiek*, pp. 7–22. Almere: LVVN.
- Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 384–391.
- Sgall, P., E. Hajičová, and J. Panevová (1986). *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.
- Sima'an, K. (1996a). Computational complexity of probabilistic disambiguation by means of tree grammar. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pp. 1175–1180.

- Sima'an, K. (1996b). An optimized algorithm for data-oriented parsing. In R. Mitkov and N. Nicolov (Eds.), *Recent Advances in Natural Language Processing. Selected Papers from RANLP '95*, pp. 35–47. John Benjamins.
- Sima'an, K. (1999). *Learning Efficient Disambiguation*. Ph. D. thesis, University of Amsterdam.
- Smith, N. A. (2006). *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph. D. thesis, Johns Hopkins University.
- Smith, N. A. and M. Johnson (2007). Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics* 33, 477–491.
- Steedman, M. (2000). *The Syntactic Process*. MIT Press.
- Steedman, M., R. Hwa, M. Osborne, and A. Sarkar (2003). Corrected co-training for statistical parsers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 95–102.
- Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics* 21, 165–202.
- Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning (2004). Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–8.
- Titov, I. and J. Henderson (2007). A latent variable model for generative dependency parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT)*, pp. 144–155.
- Toutanova, K., C. D. Manning, S. M. Shieber, D. Flickinger, and S. Oepen (2002). Parse disambiguation for a rich HPSG grammar. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT)*, pp. 253–263.
- Turian, J. and I. D. Melamed (2006). Advances in discriminative parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pp. 873–880.
- Xue, N., F. Xia, F.-D. Chiou, and M. Palmer (2004). The Penn Chinese Treebank: Phase structure annotation of a large corpus. *Journal of Natural Language Engineering* 11, 207–238.
- Yamada, H. and Y. Matsumoto (2003). Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pp. 195–206.
- Yuret, D. (1998). *Discovery of Linguistic Relations Using Lexical Attraction*. Ph. D. thesis, Massachusetts Institute of Technology.

- Zeman, D. and Z. Žabokrtský (2005). Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pp. 171–178.
- Zollmann, A. and K. Sima'an (2005). A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics* 10(2/3), 367–388.

