Structured Prediction Models via the Matrix-Tree Theorem

Terry Koo, Amir Globerson, Xavier Carreras and Michael Collins
MIT CSAIL, Cambridge, MA 02139, USA
{maestro,gamir,carreras,mcollins}@csail.mit.edu

Abstract

This paper provides an algorithmic framework for learning statistical models involving directed spanning trees, or equivalently non-projective dependency structures. We show how partition functions and marginals for directed spanning trees can be computed by an adaptation of Kirchhoff's Matrix-Tree Theorem. To demonstrate an application of the method, we perform experiments which use the algorithm in training both log-linear and max-margin dependency parsers. The new training methods give improvements in accuracy over perceptron-trained models.

1 Introduction

Learning with structured data typically involves searching or summing over a set with an exponential number of structured elements, for example the set of all parse trees for a given sentence. Methods for summing over such structures include the inside-outside algorithm for probabilistic context-free grammars (Baker, 1979), the forward-backward algorithm for hidden Markov models (Baum et al., 1970), and the belief-propagation algorithm for graphical models (Pearl, 1988). These algorithms compute marginal probabilities and partition functions, quantities which are central to many methods for the statistical modeling of complex structures (e.g., the EM algorithm (Baker, 1979; Baum et al., 1970), contrastive estimation (Smith and Eisner, 2005), training algorithms for CRFs (Lafferty et al., 2001), and training algorithms for max-margin models (Bartlett et al., 2004; Taskar et al., 2004a)).

This paper describes inside-outside-style algorithms for the case of directed spanning trees. These structures are equivalent to non-projective dependency parses (McDonald et al., 2005b), and more generally could be relevant to any task that involves learning a mapping from a graph to an underlying spanning tree. Unlike the case for projective dependency structures, partition functions and marginals for non-projective trees cannot be computed using dynamic-programming methods such as the inside-outside algorithm. In this paper we describe how these quantities can be computed by adapting a well-known result in graph theory: Kirchhoff's Matrix-Tree Theorem (Tutte, 1984). A naïve application of the theorem yields $O(n^4)$ and $O(n^6)$ algorithms for computation of the partition function and marginals, respectively. However, our adaptation finds the partition function and marginals in $O(n^3)$ time using simple matrix determinant and inversion operations.

We demonstrate an application of the new inference algorithm to non-projective dependency parsing. Specifically, we show how to implement two popular supervised learning approaches for this task: globally-normalized log-linear models and max-margin models. Log-linear estimation critically depends on the calculation of partition functions and marginals, which can be computed by our algorithms. For max-margin models, Bartlett et al. (2004) have provided a simple training algorithm, based on exponentiated-gradient (EG) updates, that requires computation of marginals and can thus be implemented within our framework. Both of these methods explicitly minimize the loss incurred when parsing the entire training set. This contrasts with the online learning algorithms used in previous work with spanning-tree models (McDonald et al., 2005b).

We applied the above two *marginal-based* training algorithms to six languages with varying degrees of non-projectivity, using datasets obtained from the CoNLL-X shared task (Buchholz and Marsi, 2006). Our experimental framework compared three training approaches: log-linear models, max-margin models, and the averaged perceptron. Each of these was applied to both projective and non-projective parsing. Our results demonstrate that marginal-based training yields models which out-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 141–150, Prague, June 2007.

perform those trained using the averaged perceptron.

In summary, the contributions of this paper are:

1. We introduce algorithms for *inside-outside-style* calculations for directed spanning trees, or equivalently non-projective dependency structures. These algorithms should have wide applicability in learning problems involving spanning-tree structures.

2. We illustrate the utility of these algorithms in log-linear training of dependency parsing models, and show improvements in accuracy when compared to averaged-perceptron training.

3. We also train max-margin models for dependency parsing via an EG algorithm (Bartlett et al., 2004). The experiments presented here constitute the first application of this algorithm to a large-scale problem. We again show improved performance over the perceptron.

The goal of our experiments is to give a rigorous comparative study of the marginal-based training algorithms and a highly-competitive baseline, the averaged perceptron, using the same feature sets for all approaches. We stress, however, that the purpose of this work is not to give competitive performance on the CoNLL data sets; this would require further engineering of the approach.

Similar adaptations of the Matrix-Tree Theorem have been developed independently and simultaneously by Smith and Smith (2007) and McDonald and Satta (2007); see Section 5 for more discussion.

2 Background

2.1 Discriminative Dependency Parsing

Dependency parsing is the task of mapping a sentence $\mathbf{x}$ to a dependency structure y. Given a sentence $\mathbf{x}$ with n words, a dependency for that sentence is a tuple (h, m) where $h \in [0 \ldots n]$ is the index of the head word in the sentence, and $m \in [1 \ldots n]$ is the index of a modifier word. The value $h = 0$ is a special *root-symbol* that may only appear as the head of a dependency. We use $\mathcal{D}(\mathbf{x})$ to refer to all possible dependencies for a sentence $\mathbf{x}$: $\mathcal{D}(\mathbf{x}) = \{(h, m) : h \in [0 \ldots n], m \in [1 \ldots n]\}$.

A dependency parse is a set of dependencies that forms a directed tree, with the sentence's root-symbol as its root. We will consider both *projective* trees, where dependencies are not allowed to cross, and *non-projective* trees, where crossing dependencies are allowed. Dependency annotations for some languages, for example Czech, can exhibit a significant number of crossing dependencies. In addition, we consider both *single-root* and *multi-root* trees. In a single-root tree y, the root-symbol has exactly one child, while in a multi-root tree, the root-symbol has one or more children. This distinction is relevant as our training sets include both single-root corpora (in which all trees are single-root structures) and multi-root corpora (in which some trees are multi-root structures).

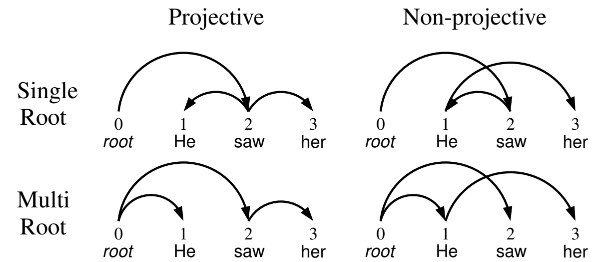

Figure 1: Examples of the four types of dependency structures. We draw dependency arcs from head to modifier.

The two distinctions described above are orthogonal, yielding four classes of dependency structures; see Figure 1 for examples of each kind of structure. We use $\mathcal{T}_p^s(\mathbf{x})$ to denote the set of all possible projective single-root dependency structures for a sentence $\mathbf{x}$, and $\mathcal{T}_{np}^s(\mathbf{x})$ to denote the set of single-root non-projective structures for $\mathbf{x}$. The sets $\mathcal{T}_p^m(\mathbf{x})$ and $\mathcal{T}_{np}^m(\mathbf{x})$ are defined analogously for multi-root structures. In contexts where any class of dependency structures may be used, we use the notation $\mathcal{T}(\mathbf{x})$ as a placeholder that may be defined as $\mathcal{T}_p^s(\mathbf{x})$, $\mathcal{T}_{np}^s(\mathbf{x})$, $\mathcal{T}_p^m(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$.

Following McDonald et al. (2005a), we use a discriminative model for dependency parsing. Features in the model are defined through a function $\mathbf{f}(\mathbf{x}, h, m)$ which maps a sentence $\mathbf{x}$ together with a dependency (h, m) to a feature vector in $\mathbb{R}^d$. A feature vector can be sensitive to any properties of the triple $(\mathbf{x}, h, m)$. Given a parameter vector $\mathbf{w}$, the optimal dependency structure for a sentence $\mathbf{x}$ is

$$y^*(\mathbf{x}; \mathbf{w}) = \underset{y \in \mathcal{T}(\mathbf{x})}{\operatorname{argmax}} \sum_{(h,m) \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m) \quad (1)$$

where the set $\mathcal{T}(\mathbf{x})$ can be defined as $\mathcal{T}_p^s(\mathbf{x})$, $\mathcal{T}_{np}^s(\mathbf{x})$, $\mathcal{T}_p^m(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$, depending on the type of parsing.

The parameters $\mathbf{w}$ will be learned from a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where each $\mathbf{x}_i$ is a sentence and each y_i is a dependency structure. Much of the previous work on learning $\mathbf{w}$ has focused on training local models (see Section 5). McDonald et al. (2005a; 2005b) trained global models using online algorithms such as the perceptron algorithm or MIRA. In this paper we consider training algorithms based on work in conditional random fields (CRFs) (Lafferty et al., 2001) and max-margin methods (Taskar et al., 2004a).

2.2 Three Inference Problems

This section highlights three inference problems which arise in training and decoding discriminative dependency parsers, and which are central to the approaches described in this paper.

Assume that we have a vector $\boldsymbol{\theta}$ with values $\theta_{h,m} \in \mathbb{R}$ for all $(h, m) \in \mathcal{D}(\mathbf{x})$; these values correspond to weights on the different dependencies in $\mathcal{D}(\mathbf{x})$. Define a conditional distribution over all dependency structures $y \in \mathcal{T}(\mathbf{x})$ as follows:

$$P(y \,|\, \mathbf{x}; \boldsymbol{\theta}) = \frac{\exp\left\{\sum_{(h,m)\in y} \theta_{h,m}\right\}}{Z(\mathbf{x}; \boldsymbol{\theta})} \quad (2)$$

$$Z(\mathbf{x}; \boldsymbol{\theta}) = \sum_{y \in \mathcal{T}(\mathbf{x})} \exp\left\{ \sum_{(h,m)\in y} \theta_{h,m} \right\} \quad (3)$$

The function $Z(\mathbf{x}; \boldsymbol{\theta})$ is commonly referred to as the *partition function*.

Given the distribution $P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$, we can define the *marginal probability* of a dependency (h, m) as

$$\mu_{h,m}(\mathbf{x}; \boldsymbol{\theta}) = \sum_{y \in \mathcal{T}(\mathbf{x}) \,:\, (h,m) \in y} P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$$

The inference problems are then as follows:

Problem 1: **Decoding:**
Find $\mathrm{argmax}_{y \in \mathcal{T}(\mathbf{x})} \sum_{(h,m)\in y} \theta_{h,m}$

Problem 2: **Computation of the Partition Function:** Calculate $Z(\mathbf{x}; \boldsymbol{\theta})$.

Problem 3: **Computation of the Marginals:**
For all $(h, m) \in \mathcal{D}(\mathbf{x})$, calculate $\mu_{h,m}(\mathbf{x}; \boldsymbol{\theta})$.

Note that all three problems require a maximization or summation over the set $\mathcal{T}(\mathbf{x})$, which is exponential in size. There is a clear motivation for being able to solve Problem 1: by setting $\theta_{h,m} = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$, the optimal dependency structure $y^*(\mathbf{x}; \mathbf{w})$ (see Eq. 1) can be computed. In this paper the motivation for solving Problems 2 and 3 arises from training algorithms for discriminative models. As we will describe in Section 4, both log-linear and max-margin models can be trained via methods that make direct use of algorithms for Problems 2 and 3.

In the case of projective dependency structures (i.e., $\mathcal{T}(\mathbf{x})$ defined as $\mathcal{T}_p^s(\mathbf{x})$ or $\mathcal{T}_p^m(\mathbf{x})$), there are well-known algorithms for all three inference problems. Decoding can be carried out using Viterbi-style dynamic-programming algorithms, for example the $O(n^3)$ algorithm of Eisner (1996). Computation of the marginals and partition function can also be achieved in $O(n^3)$ time, using a variant of the inside-outside algorithm (Baker, 1979) applied to the Eisner (1996) data structures (Paskin, 2001).

In the non-projective case (i.e., $\mathcal{T}(\mathbf{x})$ defined as $\mathcal{T}_{np}^s(\mathbf{x})$ or $\mathcal{T}_{np}^m(\mathbf{x})$), McDonald et al. (2005b) describe how the CLE algorithm (Chu and Liu, 1965; Edmonds, 1967) can be used for decoding. However, it is not possible to compute the marginals and partition function using the inside-outside algorithm. We next describe a method for computing these quantities in $O(n^3)$ time using matrix inverse and determinant operations.

3 Spanning-tree inference using the Matrix-Tree Theorem

In this section we present algorithms for computing the partition function and marginals, as defined in Section 2.2, for non-projective parsing. We first reiterate the observation of McDonald et al. (2005a) that non-projective parses correspond to directed spanning trees on a complete directed graph of n nodes, where n is the length of the sentence. The above inference problems thus involve summation over the set of all directed spanning trees. Note that this set is exponentially large, and there is no obvious method for decomposing the sum into dynamic-programming-like subproblems. This section describes how a variant of Kirchhoff's Matrix-Tree Theorem (Tutte, 1984) can be used to evaluate the partition function and marginals efficiently.

In what follows, we consider the single-root setting (i.e., $\mathcal{T}(\mathbf{x}) = \mathcal{T}_{np}^s(\mathbf{x})$), leaving the multi-root case (i.e., $\mathcal{T}(\mathbf{x}) = \mathcal{T}_{np}^{m}(\mathbf{x})$) to Section 3.3. For a sentence $\mathbf{x}$ with n words, define a complete directed graph G on n nodes, where each node corresponds to a word in $\mathbf{x}$, and each edge corresponds to a dependency between two words in $\mathbf{x}$. Note that G does *not* include the root-symbol $h = 0$, nor does it account for any dependencies $(0, m)$ headed by the root-symbol. We assign non-negative weights to the edges of this graph, yielding the following weighted adjacency matrix $A(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n}$, for $h, m = 1 \ldots n$:

$$A_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} 0, & \text{if } h = m \\ \exp\{\theta_{h,m}\}, & \text{otherwise} \end{cases}$$

To account for the dependencies $(0, m)$ headed by the root-symbol, we define a vector of root-selection scores $\mathbf{r}(\boldsymbol{\theta}) \in \mathbb{R}^n$, for $m = 1 \ldots n$:

$$r_m(\boldsymbol{\theta}) \;=\; \exp\{\theta_{0,m}\}$$

Let the weight of a dependency structure $y \in \mathcal{T}_{np}^{s}(\mathbf{x})$ be defined as:

$$\psi(y;\boldsymbol{\theta}) \;=\; r_{\mathrm{root}(y)}(\boldsymbol{\theta}) \prod_{(h,m)\in y\,:\, h\neq 0} A_{h,m}(\boldsymbol{\theta})$$

Here, $\mathrm{root}(y) = m : (0, m) \in y$ is the child of the root-symbol; there is exactly one such child, since $y \in \mathcal{T}_{np}^{s}(\mathbf{x})$. Eq. 2 and 3 can be rephrased as:

$$P(y \,|\, \mathbf{x};\boldsymbol{\theta}) \;=\; \frac{\psi(y;\boldsymbol{\theta})}{Z(\mathbf{x};\boldsymbol{\theta})} \qquad (4)$$

$$Z(\mathbf{x};\boldsymbol{\theta}) \;=\; \sum_{y\in\mathcal{T}_{np}^{s}(\mathbf{x})} \psi(y;\boldsymbol{\theta}) \qquad (5)$$

In the remainder of this section, we drop the notational dependence on $\mathbf{x}$ for brevity.

The original Matrix-Tree Theorem addressed the problem of counting the number of undirected spanning trees in an undirected graph. For the models we study here, we require a sum of *weighted* and *directed* spanning trees. Tutte (1984) extended the Matrix-Tree Theorem to this case. We briefly summarize his method below.

First, define the Laplacian matrix $L(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n}$ of G, for $h, m = 1 \ldots n$:

$$L_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} \sum_{h'=1}^{n} A_{h',m}(\boldsymbol{\theta}) & \text{if } h = m \\ -A_{h,m}(\boldsymbol{\theta}) & \text{otherwise} \end{cases}$$

Second, for a matrix X, let $X^{(h,m)}$ be the *minor* of X with respect to row h and column m; i.e., the determinant of the matrix formed by deleting row h and column m from X. Finally, define the weight of any directed spanning tree of G to be the product of the weights $A_{h,m}(\boldsymbol{\theta})$ for the edges in that tree.

Theorem 1 *(Tutte, 1984, p. 140). Let $L(\boldsymbol{\theta})$ be the Laplacian matrix of G. Then $L^{(m,m)}(\boldsymbol{\theta})$ is equal to the sum of the weights of all directed spanning trees of G which are rooted at m. Furthermore, the minors vary only in sign when traversing the columns of the Laplacian (Tutte, 1984, p. 150):*

$$\forall h, m\colon\; (-1)^{h+m} L^{(h,m)}(\boldsymbol{\theta}) \;=\; L^{(m,m)}(\boldsymbol{\theta}) \qquad (6)$$

3.1 Partition functions via matrix determinants

From Theorem 1, it directly follows that

$$L^{(m,m)}(\boldsymbol{\theta}) \;=\; \sum_{y\in\mathcal{U}(m)} \;\prod_{(h,m)\in y\,:\, h\neq 0} A_{h,m}(\boldsymbol{\theta})$$

where $\mathcal{U}(m) = \{y \in \mathcal{T}_{np}^{s} : \mathrm{root}(y) = m\}$. A naïve method for computing the partition function is therefore to evaluate

$$Z(\boldsymbol{\theta}) \;=\; \sum_{m=1}^{n} r_m(\boldsymbol{\theta}) L^{(m,m)}(\boldsymbol{\theta})$$

The above would require calculating n determinants, resulting in $O(n^4)$ complexity. However, as we show below $Z(\boldsymbol{\theta})$ may be obtained in $O(n^3)$ time using a single determinant evaluation.

Define a new matrix $\hat{L}(\boldsymbol{\theta})$ to be $L(\boldsymbol{\theta})$ with the first row replaced by the root-selection scores:

$$\hat{L}_{h,m}(\boldsymbol{\theta}) \;=\; \begin{cases} r_m(\boldsymbol{\theta}) & h = 1 \\ L_{h,m}(\boldsymbol{\theta}) & h > 1 \end{cases}$$

This matrix allows direct computation of the partition function, as the following proposition shows.

Proposition 1 *The partition function in Eq. 5 is given by $Z(\boldsymbol{\theta}) = |\hat{L}(\boldsymbol{\theta})|$.*

Proof: *Consider the row expansion of $|\hat{L}(\boldsymbol{\theta})|$ with respect to row 1:*

$$\begin{aligned} |\hat{L}(\boldsymbol{\theta})| &= \sum_{m=1}^{n} (-1)^{1+m} \hat{L}_{1,m}(\boldsymbol{\theta}) \hat{L}^{(1,m)}(\boldsymbol{\theta}) \\ &= \sum_{m=1}^{n} (-1)^{1+m} r_m(\boldsymbol{\theta}) L^{(1,m)}(\boldsymbol{\theta}) \\ &= \sum_{m=1}^{n} r_m(\boldsymbol{\theta}) L^{(m,m)}(\boldsymbol{\theta}) \;=\; Z(\boldsymbol{\theta}) \end{aligned}$$

The second line follows from the construction of $\hat{L}(\boldsymbol{\theta})$, and the third line follows from Eq. 6. ■

3.2 Marginals via matrix inversion

The marginals we require are given by

$$\mu_{h,m}(\boldsymbol{\theta}) \;=\; \frac{1}{Z(\boldsymbol{\theta})} \sum_{y \in \mathcal{T}^s_{np} \,:\, (h,m) \in y} \psi(y; \boldsymbol{\theta})$$

To calculate these marginals efficiently for all values of (h, m) we use a well-known identity relating the log partition-function to marginals

$$\mu_{h,m}(\boldsymbol{\theta}) \;=\; \frac{\partial \log Z(\boldsymbol{\theta})}{\partial \theta_{h,m}}$$

Since the partition function in this case has a closed-form expression (i.e., the determinant of a matrix constructed from $\boldsymbol{\theta}$), the marginals can also obtained in closed form. Using the chain rule, the derivative of the log partition-function in Proposition 1 is

$$\begin{aligned}\mu_{h,m}(\boldsymbol{\theta}) \;&=\; \frac{\partial \log |\hat{L}(\boldsymbol{\theta})|}{\partial \theta_{h,m}} \\ &=\; \sum_{h'=1}^{n} \sum_{m'=1}^{n} \frac{\partial \log |\hat{L}(\boldsymbol{\theta})|}{\partial \hat{L}_{h',m'}(\boldsymbol{\theta})} \frac{\partial \hat{L}_{h',m'}(\boldsymbol{\theta})}{\partial \theta_{h,m}}\end{aligned}$$

To perform the derivative, we use the identity

$$\frac{\partial \log |X|}{\partial X} \;=\; \left(X^{-1}\right)^T$$

and the fact that $\partial \hat{L}_{h',m'}(\boldsymbol{\theta}) / \partial \theta_{h,m}$ is nonzero for only a few h', m'. Specifically, when $h = 0$, the marginals are given by

$$\mu_{0,m}(\boldsymbol{\theta}) \;=\; r_m(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,1}$$

and for $h > 0$, the marginals are given by

$$\begin{aligned}\mu_{h,m}(\boldsymbol{\theta}) \;=\;& (1 - \delta_{1,m}) A_{h,m}(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,m} - \\ & (1 - \delta_{h,1}) A_{h,m}(\boldsymbol{\theta}) \left[\hat{L}^{-1}(\boldsymbol{\theta})\right]_{m,h}\end{aligned}$$

where $\delta_{h,m}$ is the Kronecker delta. Thus, the complexity of evaluating *all* the relevant marginals is dominated by the matrix inversion, and the total complexity is therefore $O(n^3)$.

3.3 Multiple Roots

In the case of multiple roots, we can still compute the partition function and marginals efficiently. In fact, the derivation of this case is simpler than for single-root structures. Create an extended graph G' which augments G with a dummy root node that has edges pointing to all of the existing nodes, weighted by the appropriate root-selection scores. Note that there is a bijection between directed spanning trees of G' rooted at the dummy root and multi-root structures $y \in \mathcal{T}^m_{np}(\mathbf{x})$. Thus, Theorem 1 can be used to compute the partition function directly: construct a Laplacian matrix $L(\boldsymbol{\theta})$ for G' and compute the minor $L^{(0,0)}(\boldsymbol{\theta})$. Since this minor is also a determinant, the marginals can be obtained analogously to the single-root case. More concretely, this technique corresponds to defining the matrix $\hat{L}(\boldsymbol{\theta})$ as

$$\hat{L}(\boldsymbol{\theta}) \;=\; L(\boldsymbol{\theta}) + \operatorname{diag}(\mathbf{r}(\boldsymbol{\theta}))$$

where $\operatorname{diag}(\mathbf{v})$ is the diagonal matrix with the vector $\mathbf{v}$ on its diagonal.

3.4 Labeled Trees

The techniques above extend easily to the case where dependencies are labeled. For a model with L different labels, it suffices to define the edge and root scores as $A_{h,m}(\boldsymbol{\theta}) = \sum_{\ell=1}^{L} \exp\{\theta_{h,m,\ell}\}$ and $r_m(\boldsymbol{\theta}) = \sum_{\ell=1}^{L} \exp\{\theta_{0,m,\ell}\}$. The partition function over labeled trees is obtained by operating on these values as described previously, and the marginals are given by an application of the chain rule. Both inference problems are solvable in $O(n^3 + Ln^2)$ time.

4 Training Algorithms

This section describes two methods for parameter estimation that rely explicitly on the computation of the partition function and marginals.

4.1 Log-Linear Estimation

In conditional log-linear models (Johnson et al., 1999; Lafferty et al., 2001), a distribution over parse trees for a sentence $\mathbf{x}$ is defined as follows:

$$P(y \,|\, \mathbf{x}; \mathbf{w}) \;=\; \frac{\exp\left\{\sum_{(h,m) \in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)\right\}}{Z(\mathbf{x}; \mathbf{w})} \qquad (7)$$

where $Z(\mathbf{x}; \mathbf{w})$ is the partition function, a sum over $\mathcal{T}^s_p(\mathbf{x})$, $\mathcal{T}^s_{np}(\mathbf{x})$, $\mathcal{T}^m_p(\mathbf{x})$ or $\mathcal{T}^m_{np}(\mathbf{x})$.

We train the model using the approach described by Sha and Pereira (2003). Assume that we have a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. The optimal parameters are taken to be $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} L(\mathbf{w})$ where

$$L(\mathbf{w}) = -C \sum_{i=1}^{N} \log P(y_i \,|\, \mathbf{x}_i; \mathbf{w}) + \frac{1}{2}||\mathbf{w}||^2$$

The parameter $C > 0$ is a constant dictating the level of regularization in the model.

Since $L(\mathbf{w})$ is a convex function, gradient descent methods can be used to search for the global minimum. Such methods typically involve repeated computation of the loss $L(\mathbf{w})$ and gradient $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$, requiring efficient implementations of both functions. Note that the log-probability of a parse is

$$\log P(y \,|\, \mathbf{x}; \mathbf{w}) = \sum_{(h,m)\in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m) - \log Z(\mathbf{x}; \mathbf{w})$$

so that the main issue in calculating the loss function $L(\mathbf{w})$ is the evaluation of the partition functions $Z(\mathbf{x}_i; \mathbf{w})$. The gradient of the loss is given by

$$\begin{aligned}\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} &= \mathbf{w} - C \sum_{i=1}^{N} \sum_{(h,m)\in y_i} \mathbf{f}(\mathbf{x}_i, h, m) \\ &+ C \sum_{i=1}^{N} \sum_{(h,m)\in \mathcal{D}(\mathbf{x}_i)} \mu_{h,m}(\mathbf{x}_i; \mathbf{w}) \mathbf{f}(\mathbf{x}_i, h, m)\end{aligned}$$

where

$$\mu_{h,m}(\mathbf{x}; \mathbf{w}) = \sum_{y\in\mathcal{T}(\mathbf{x})\,:\,(h,m)\in y} P(y \,|\, \mathbf{x}; \mathbf{w})$$

is the marginal probability of a dependency (h, m). Thus, the main issue in the evaluation of the gradient is the computation of the marginals $\mu_{h,m}(\mathbf{x}_i; \mathbf{w})$.

Note that Eq. 7 forms a special case of the log-linear distribution defined in Eq. 2 in Section 2.2. If we set $\theta_{h,m} = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, h, m)$ then we have $P(y \,|\, \mathbf{x}; \mathbf{w}) = P(y \,|\, \mathbf{x}; \boldsymbol{\theta})$, $Z(\mathbf{x}; \mathbf{w}) = Z(\mathbf{x}; \boldsymbol{\theta})$, and $\mu_{h,m}(\mathbf{x}; \mathbf{w}) = \mu_{h,m}(\mathbf{x}; \boldsymbol{\theta})$. Thus in the projective case the inside-outside algorithm can be used to calculate the partition function and marginals, thereby enabling training of a log-linear model; in the non-projective case the algorithms in Section 3 can be used for this purpose.

4.2 Max-Margin Estimation

The second learning algorithm we consider is the large-margin approach for structured prediction (Taskar et al., 2004a; Taskar et al., 2004b). Learning in this framework again involves minimization of a convex function $L(\mathbf{w})$. Let the *margin* for parse tree y on the i'th training example be defined as

$$m_{i,y}(\mathbf{w}) = \sum_{(h,m)\in y_i} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, h, m) - \sum_{(h,m)\in y} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, h, m)$$

The loss function is then defined as

$$L(\mathbf{w}) = C \sum_{i=1}^{N} \max_{y\in\mathcal{T}(\mathbf{x}_i)} (E_{i,y} - m_{i,y}(\mathbf{w})) + \frac{1}{2}||\mathbf{w}||^2$$

where $E_{i,y}$ is a measure of the loss—or number of errors—for parse y on the i'th training sentence. In this paper we take $E_{i,y}$ to be the number of incorrect dependencies in the parse tree y when compared to the gold-standard parse tree y_i.

The definition of $L(\mathbf{w})$ makes use of the expression $\max_{y\in\mathcal{T}(\mathbf{x}_i)} (E_{i,y} - m_{i,y}(\mathbf{w}))$ for the i'th training example, which is commonly referred to as the *hinge loss*. Note that $E_{i,y_i} = 0$, and also that $m_{i,y_i}(\mathbf{w}) = 0$, so that the hinge loss is always non-negative. In addition, the hinge loss is 0 if and only if $m_{i,y}(\mathbf{w}) \geq E_{i,y}$ for all $y \in \mathcal{T}(\mathbf{x}_i)$. Thus the hinge loss directly penalizes margins $m_{i,y}(\mathbf{w})$ which are less than their corresponding losses $E_{i,y}$.

Figure 2 shows an algorithm for minimizing $L(\mathbf{w})$ that is based on the exponentiated-gradient algorithm for large-margin optimization described by Bartlett et al. (2004). The algorithm maintains a set of weights $\theta_{i,h,m}$ for $i = 1 \ldots N, (h, m) \in \mathcal{D}(\mathbf{x}_i)$, which are updated example-by-example. The algorithm relies on the repeated computation of marginal values $\mu_{i,h,m}$, which are defined as follows:[1]

$$\mu_{i,h,m} = \sum_{y\in\mathcal{T}(\mathbf{x}_i)\,:\,(h,m)\in y} P(y \,|\, \mathbf{x}_i) \tag{8}$$

$$P(y \,|\, \mathbf{x}_i) = \frac{\exp\left\{\sum_{(h,m)\in y} \theta_{i,h,m}\right\}}{\sum_{y'\in\mathcal{T}(\mathbf{x}_i)} \exp\left\{\sum_{(h,m)\in y'} \theta_{i,h,m}\right\}}$$

A similar definition is used to derive marginal values $\mu'_{i,h,m}$ from the values $\theta'_{i,h,m}$. Computation of the μ and μ' values is again inference of the form described in Problem 3 in Section 2.2, and can be

[1] Bartlett et al. (2004) write $P(y \,|\, \mathbf{x}_i)$ as $\alpha_{i,y}$. The $\alpha_{i,y}$ variables are dual variables that appear in the dual objective function, i.e., the convex dual of $L(\mathbf{w})$. Analysis of the algorithm shows that as the $\theta_{i,h,m}$ variables are updated, the dual variables converge to the optimal point of the dual objective, and the parameters $\mathbf{w}$ converge to the minimum of $L(\mathbf{w})$.

Inputs: Training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

Parameters: Regularization constant C, starting point β, number of passes over training set T.

Data Structures: Real values $\theta_{i,h,m}$ and $l_{i,h,m}$ for $i = 1 \ldots N, (h,m) \in \mathcal{D}(\mathbf{x}_i)$. Learning rate η.

Initialization: Set learning rate $\eta = \frac{1}{C}$. Set $\theta_{i,h,m} = \beta$ for $(h,m) \in y_i$, and $\theta_{i,h,m} = 0$ for $(h,m) \notin y_i$. Set $l_{i,h,m} = 0$ for $(h,m) \in y_i$, and $l_{i,h,m} = 1$ for $(h,m) \notin y_i$. Calculate initial parameters as

$$\mathbf{w} = C \sum_i \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} \delta_{i,h,m} \mathbf{f}(\mathbf{x}_i, h, m)$$

where $\delta_{i,h,m} = (1 - l_{i,h,m} - \mu_{i,h,m})$ and the $\mu_{i,h,m}$ values are calculated from the $\theta_{i,h,m}$ values as described in Eq. 8.

Algorithm: Repeat T passes over the training set, where each pass is as follows:

Set $obj = 0$

For $i = 1 \ldots N$

- For all $(h,m) \in \mathcal{D}(\mathbf{x}_i)$:
  $\theta'_{i,h,m} = \theta_{i,h,m} + \eta C \left(l_{i,h,m} + \mathbf{w} \cdot \mathbf{f}(\mathbf{x}_i, h, m) \right)$
- For example i, calculate marginals $\mu_{i,h,m}$ from $\theta_{i,h,m}$ values, and marginals $\mu'_{i,h,m}$ from $\theta'_{i,h,m}$ values (see Eq. 8).
- Update the parameters:
  $\mathbf{w} = \mathbf{w} + C \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} \delta_{i,h,m} \mathbf{f}(\mathbf{x}_i, h, m)$
  where $\delta_{i,h,m} = \mu_{i,h,m} - \mu'_{i,h,m}$,
- For all $(h,m) \in \mathcal{D}(\mathbf{x}_i)$, set $\theta_{i,h,m} = \theta'_{i,h,m}$
- Set $obj = obj + C \sum_{(h,m) \in \mathcal{D}(\mathbf{x}_i)} l_{i,h,m} \mu'_{i,h,m}$

Set $obj = obj - \frac{||\mathbf{w}||^2}{2}$. If obj has decreased compared to last iteration, set $\eta = \frac{\eta}{2}$.

Output: Parameter values $\mathbf{w}$.

Figure 2: The EG Algorithm for Max-Margin Estimation. The learning rate η is halved each time the dual objective function (see (Bartlett et al., 2004)) fails to increase. In our experiments we chose $\beta = 9$, which was found to work well during development of the algorithm.

achieved using the inside-outside algorithm for projective structures, and the algorithms described in Section 3 for non-projective structures.

5 Related Work

Global log-linear training has been used in the context of PCFG parsing (Johnson, 2001). Riezler et al. (2004) explore a similar application of log-linear models to LFG parsing. Max-margin learning has been applied to PCFG parsing by Taskar et al. (2004b). They show that this problem has a QP dual of polynomial size, where the dual variables correspond to marginal probabilities of CFG rules. A similar QP dual may be obtained for max-margin projective dependency parsing. However, for non-projective parsing, the dual QP would require an exponential number of constraints on the dependency marginals (Chopra, 1989). Nevertheless, alternative optimization methods like that of Tsochantaridis et al. (2004), or the EG method presented here, can still be applied.

The majority of previous work on dependency parsing has focused on local (i.e., classification of individual edges) discriminative training methods (Yamada and Matsumoto, 2003; Nivre et al., 2004; Y. Cheng, 2005). Non-local (i.e., classification of entire trees) training methods were used by McDonald et al. (2005a), who employed online learning.

Dependency parsing accuracy can be improved by allowing second-order features, which consider more than one dependency simultaneously. McDonald and Pereira (2006) define a second-order dependency parsing model in which interactions between adjacent siblings are allowed, and Carreras (2007) defines a second-order model that allows grandparent and sibling interactions. Both authors give polytime algorithms for exact projective parsing. By adapting the inside-outside algorithm to these models, partition functions and marginals can be computed for second-order projective structures, allowing log-linear and max-margin training to be applied via the framework developed in this paper. For higher-order non-projective parsing, however, computational complexity results (McDonald and Pereira, 2006; McDonald and Satta, 2007) indicate that exact solutions to the three inference problems of Section 2.2 will be intractable. Exploration of approximate second-order non-projective inference is a natural avenue for future research.

Two other groups of authors have independently and simultaneously proposed adaptations of the Matrix-Tree Theorem for structured inference on directed spanning trees (McDonald and Satta, 2007; Smith and Smith, 2007). There are some algorithmic differences between these papers and ours. First, we define both multi-root and single-root algorithms, whereas the other papers only consider multi-root parsing. This distinction can be important as one often expects a dependency structure to have exactly one child attached to the root-symbol, as is the case in a single-root structure. Second, McDonald and Satta (2007) propose an $O(n^5)$ algorithm for computing the marginals, as opposed to the $O(n^3)$ matrix-inversion approach used by Smith and Smith (2007) and ourselves.

In addition to the algorithmic differences, both groups of authors consider applications of the Matrix-Tree Theorem which we have not discussed. For example, both papers propose minimum-risk decoding, and McDonald and Satta (2007) discuss unsupervised learning and language modeling, while Smith and Smith (2007) define hidden-variable models based on spanning trees.

In this paper we used EG training methods only for max-margin models (Bartlett et al., 2004). However, Globerson et al. (2007) have recently shown how EG updates can be applied to efficient training of log-linear models.

6 Experiments on Dependency Parsing

In this section, we present experimental results applying our inference algorithms for dependency parsing models. Our primary purpose is to establish comparisons along two relevant dimensions: projective training vs. non-projective training, and marginal-based training algorithms vs. the averaged perceptron. The feature representation and other relevant dimensions are kept fixed in the experiments.

6.1 Data Sets and Features

We used data from the CoNLL-X shared task on multilingual dependency parsing (Buchholz and Marsi, 2006). In our experiments, we used a subset consisting of six languages; Table 1 gives details of the data sets used.[2] For each language we created a validation set that was a subset of the CoNLL-X training set for that language. The remainder of each training set was used to train the models for the different languages. The validation sets were used to tune the meta-parameters (e.g., the value of the regularization constant C) of the different training algorithms. We used the official test sets and evaluation script from the CoNLL-X task. All of the results that we report are for unlabeled dependency parsing.[3]

language	%cd	train	val.	test
Arabic	0.34	49,064	5,315	5,373
Dutch	4.93	178,861	16,208	5,585
Japanese	0.70	141,966	9,495	5,711
Slovene	1.59	22,949	5,801	6,390
Spanish	0.06	78,310	11,024	5,694
Turkish	1.26	51,827	5,683	7,547

Table 1: Information for the languages in our experiments. The 2nd column (%cd) is the percentage of crossing dependencies in the training and validation sets. The last three columns report the size in tokens of the training, validation and test sets.

The non-projective models were trained on the CoNLL-X data in its original form. Since the projective models assume that the dependencies in the data are non-crossing, we created a second training set for each language where non-projective dependency structures were automatically transformed into projective structures. All projective models were trained on these new training sets.[4] Our feature space is based on that of McDonald et al. (2005a).[5]

6.2 Results

We performed experiments using three training algorithms: the averaged perceptron (Collins, 2002), log-linear training (via conjugate gradient descent), and max-margin training (via the EG algorithm). Each of these algorithms was trained using projective and non-projective methods, yielding six training settings per language. The different training algorithms have various meta-parameters, which we optimized on the validation set for each language/training-setting combination. The

[2]Our subset includes the two languages with the lowest accuracy in the CoNLL-X evaluations (Turkish and Arabic), the language with the highest accuracy (Japanese), the most non-projective language (Dutch), a moderately non-projective language (Slovene), and a highly projective language (Spanish). All languages but Spanish have multi-root parses in their data. We are grateful to the providers of the treebanks that constituted the data of our experiments (Hajič et al., 2004; van der Beek et al., 2002; Kawata and Bartels, 2000; Džeroski et al., 2006; Civit and Martí, 2002; Oflazer et al., 2003).

[3]Our algorithms also support labeled parsing (see Section 3.4). Initial experiments with labeled models showed the same trend that we report here for unlabeled parsing, so for simplicity we conducted extensive experiments only for unlabeled parsing.

[4]The transformations were performed by running the projective parser with score +1 on correct dependencies and -1 otherwise: the resulting trees are guaranteed to be projective and to have a minimum loss with respect to the correct tree. Note that only the training sets were transformed.

[5]It should be noted that McDonald et al. (2006) use a richer feature set that is incomparable to our features.

	Perceptron		Max-Margin		Log-Linear	
	p	np	p	np	p	np
Ara	71.74	71.84	71.74	72.99	73.11	**73.67**
Dut	77.17	78.83	76.53	**79.69**	76.23	79.55
Jap	91.90	91.78	92.10	**92.18**	91.68	91.49
Slo	78.02	78.66	79.78	**80.10**	78.24	79.66
Spa	81.19	80.02	81.71	**81.93**	81.75	81.57
Tur	71.22	71.70	**72.83**	72.02	72.26	72.62

Table 2: Test data results. The p and np columns show results with projective and non-projective training respectively.

	Ara	Dut	Jap	Slo	Spa	Tur	AV
P	71.74	78.83	91.78	78.66	81.19	71.70	79.05
E	72.99	**79.69**	**92.18**	**80.10**	**81.93**	72.02	**79.82**
L	**73.67**	79.55	91.49	79.66	81.57	**72.26**	79.71

Table 3: Results for the three training algorithms on the different languages (P = perceptron, E = EG, L = log-linear models). AV is an average across the results for the different languages.

averaged perceptron has a single meta-parameter, namely the number of iterations over the training set. The log-linear models have two meta-parameters: the regularization constant C and the number of gradient steps T taken by the conjugate-gradient optimizer. The EG approach also has two meta-parameters: the regularization constant C and the number of iterations, T.[6] For models trained using non-projective algorithms, both projective and non-projective parsing was tested on the validation set, and the highest scoring of these two approaches was then used to decode test data sentences.

Table 2 reports test results for the six training scenarios. These results show that for Dutch, which is the language in our data that has the highest number of crossing dependencies, non-projective training gives significant gains over projective training for all three training methods. For the other languages, non-projective training gives similar or even improved performance over projective training.

Table 3 gives an additional set of results, which were calculated as follows. For each of the three training methods, we used the validation set results to choose between projective and non-projective training. This allows us to make a direct comparison of the three training algorithms. Table 3 shows the results of this comparison.[7] The results show that log-linear and max-margin models both give a higher average accuracy than the perceptron. For some languages (e.g., Japanese), the differences from the perceptron are small; however for other languages (e.g., Arabic, Dutch or Slovene) the improvements seen are quite substantial.

7 Conclusions

This paper describes inference algorithms for spanning-tree distributions, focusing on the fundamental problems of computing partition functions and marginals. Although we concentrate on log-linear and max-margin estimation, the inference algorithms we present can serve as black-boxes in many other statistical modeling techniques.

Our experiments suggest that marginal-based training produces more accurate models than perceptron learning. Notably, this is the first large-scale application of the EG algorithm, and shows that it is a promising approach for structured learning.

In line with McDonald et al. (2005b), we confirm that spanning-tree models are well-suited to dependency parsing, especially for highly non-projective languages such as Dutch. Moreover, spanning-tree models should be useful for a variety of other problems involving structured data.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments. In addition, the authors gratefully acknowledge the following sources of support. Terry Koo was funded by a grant from the NSF (DMS-0434222) and a grant from NTT, Agmt. Dtd. 6/21/1998. Amir Globerson was supported by a fellowship from the Rothschild Foundation - Yad Hanadiv. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222.

[6] We trained the perceptron for 100 iterations, and chose the iteration which led to the best score on the validation set. Note that in all of our experiments, the best perceptron results were actually obtained with 30 or fewer iterations. For the log-linear and EG algorithms we tested a number of values for C, and for each value of C ran 100 gradient steps or EG iterations, finally choosing the best combination of C and T found in validation.

[7] We ran the sign test at the sentence level to measure the statistical significance of the results aggregated across the six languages. Out of 2,472 sentences total, log-linear models gave improved parses over the perceptron on 448 sentences, and worse parses on 343 sentences. The max-margin method gave improved/worse parses for 500/383 sentences. Both results are significant with $p \leq 0.001$.

References

J. Baker. 1979. Trainable grammars for speech recognition. In *97th meeting of the Acoustical Society of America*.

P. Bartlett, M. Collins, B. Taskar, and D. McAllester. 2004. Exponentiated gradient algorithms for large–margin structured classification. In *NIPS*.

L.E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41:164–171.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. CoNLL-X*.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. EMNLP-CoNLL*.

S. Chopra. 1989. On the spanning tree polyhedron. *Oper. Res. Lett.*, pages 25–29.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Civit and M[a] A. Martí. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.

A. Globerson, T. Koo, X. Carreras, and M. Collins. 2007. Exponentiated gradient algorithms for log-linear structured prediction. In *Proc. ICML*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proc. ACL*.

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *Proc. ACL*.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditonal random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*.

R. McDonald and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. IWPT*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT-EMNLP*.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency parsing with a two-stage discriminative parser. In *Proc. CoNLL-X*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. CoNLL*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, chapter 15. Kluwer Academic Publishers.

M.A. Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical Report UCB/CSD-01-1148, University of California, Berkeley.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd edition)*. Morgan Kaufmann Publishers.

S. Riezler, R. Kaplan, T. King, J. Maxwell, A. Vasserman, and R. Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc. HLT-NAACL*.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*.

N.A. Smith and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. ACL*.

D.A. Smith and N.A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proc. EMNLP-CoNLL*.

B. Taskar, C. Guestrin, and D. Koller. 2004a. Max-margin markov networks. In *NIPS*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004b. Max-margin parsing. In *Proc. EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*.

W. Tutte. 1984. *Graph Theory*. Addison-Wesley.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Y. Matsumoto Y. Cheng, M. Asahara. 2005. Machine learning-based dependency analyzer for chinese. In *Proc. ICCC*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.