Probabilistic Models of Nonprojective Dependency Trees

David A. Smith
Department of Computer Science
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218 USA
dasmith@cs.jhu.edu

Noah A. Smith
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
nasmith@cs.cmu.edu

Abstract

A notable gap in research on statistical dependency parsing is a proper conditional probability distribution over nonprojective dependency trees for a given sentence. We exploit the Matrix Tree Theorem (Tutte, 1984) to derive an algorithm that efficiently sums the scores of all nonprojective trees in a sentence, permitting the definition of a conditional log-linear model over trees. While discriminative methods, such as those presented in McDonald et al. (2005b), obtain very high accuracy on standard dependency parsing tasks and can be trained and applied without marginalization, "summing trees" permits some alternative techniques of interest. Using the summing algorithm, we present competitive experimental results on four nonprojective languages, for maximum conditional likelihood estimation, minimum Bayes-risk parsing, and hidden variable training.

1 Introduction

Recently dependency parsing has received renewed interest, both in the parsing literature (Buchholz and Marsi, 2006) and in applications like translation (Quirk et al., 2005) and information extraction (Culotta and Sorensen, 2004). Dependency parsing can be used to provide a "bare bones" syntactic structure that approximates semantics, and it has the additional advantage of admitting fast parsing algorithms (Eisner, 1996; McDonald et al., 2005b) with a negligible grammar constant in many cases.

The latest state-of-the-art statistical dependency parsers are **discriminative**, meaning that they are based on classifiers trained to score trees, given a sentence, either via factored whole-structure scores (McDonald et al., 2005a) or local parsing decision scores (Hall et al., 2006). In the works cited, these scores are not intended to be interpreted as probabilistic quantities.

Here we consider weighted dependency parsing models that can be used to define well-formed conditional distributions $p(\mathbf{y} \mid \mathbf{x})$, for dependency trees $\mathbf{y}$ and a sentence $\mathbf{x}$. Conditional distributions over outputs (here, trees) given inputs (here, sentences) have certain advantages. They permit marginalization over trees to compute **posteriors** of interesting sub-events (e.g., the probability that two noun tokens bear a relation, regardless of which tree is correct). A probability model permits alternative **decoding** procedures (Goodman, 1996). Well-motivated **hidden variable** training procedures (such as EM and conditional EM) are also readily available for probabilistic models. Finally, probability models can be chained together (as in a noisy channel model), mixed, or combined in a product-of-experts.

Sequence models, context-free models, and dependency models have appeared in several guises; a cross-model comparison clarifies the contribution of this paper. First, there were generative, stochastic models like HMMs, PCFGs, and Eisner's (1996) models. Local discriminative classifiers were proposed by McCallum et al. (2000) for sequence modeling, by Ratnaparkhi et al. (1994) for constituent parsing, and by Hall et al. (2006) (among others) for

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 132–140, Prague, June 2007.

dependencies. Large-margin whole-structure models were proposed for sequence labeling by Altun et al. (2003), for constituents by Taskar et al. (2004), and for dependency trees by McDonald et al. (2005a). In this paper, we propose a model most similar to the conditional random fields—interpretable as log-linear models—of Lafferty et al. (2001), which are now widely used for sequence labeling. Log-linear models have been used in parsing by Riezler et al. (2000) (for constraint-based grammars) and Johnson (2001) and Miyao and Tsujii (2002) (for CFGs). Like McDonald et al., we use an edge-factored model that permits *nonprojective* trees; like Lafferty et al., we argue for an alternative interpretation as a **log-linear model** over structures, conditioned on the observed sentence.

In Section 2 we point out what would be required, computationally, for conditional training of nonprojective dependency models. The solution to the conditionalization problem is given in Section 3, using a widely-known but newly-applied Matrix Tree Theorem due to Tutte (1984), and experimental results are presented with a comparison to the MIRA learning algorithm used by McDonald et al. (2005a). We go on to describe and experiment with two useful applications of conditional modeling: minimum Bayes-risk decoding (Section 4) and hidden-variable training by conditional maximum likelihood estimation (Section 5). Discussion in Section 6 considers the implications of our experimental results.

Two indepedent papers, published concurrently with this one, report closely related results to ours. Koo et al. (2007) and McDonald and Satta (2007) both describe how the Matrix Tree Theorem can be applied to computing the sum of scores of edge-factored dependency trees and the edge marginals. Koo et al. compare conditional likelihood training (as here) to the averaged perceptron and a maximum margin model trained using exponentiated-gradient (Bartlett et al., 2004); the latter requires the same marginalization calculations as conditional log-linear estimation. McDonald and Satta discuss a variety of applications (including minimum Bayes-risk decoding) and give complexity results for non-edge-factored models. Interested readers are referred to those papers for further discussion.

2 Conditional Training for Nonprojective Dependency Models

Let $\mathbf{x} = \langle x_1, ..., x_n \rangle$ be a sequence of words (possibly with POS tags, lemmas, and morphological information) that are the input to a parser. $\mathbf{y}$ will refer to a directed, unlabeled dependency tree, which is a map $\mathbf{y} : \{1, ..., n\} \rightarrow \{0, ..., n\}$ from child indices to parent indices; x_0 is the invisible "wall" symbol. Let $\mathcal{Y}_\mathbf{x}$ be the set of valid dependency trees for $\mathbf{x}$. In this paper, $\mathcal{Y}_\mathbf{x}$ is equivalent to the set of all directed spanning trees over $\mathbf{x}$.[1]

A conditional model defines a family of probability distributions $p(\mathbf{y} \mid \mathbf{x})$, for all $\mathbf{x}$ and $\mathbf{y} \in \mathcal{Y}_\mathbf{x}$. We propose that this model take a log-linear form:

$$p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) = \frac{e^{\vec{\theta} \cdot \vec{f}(\mathbf{x}, \mathbf{y})}}{\sum_{\mathbf{y}' \in \mathcal{Y}_\mathbf{x}} e^{\vec{\theta} \cdot \vec{f}(\mathbf{x}, \mathbf{y}')}} = \frac{e^{\vec{\theta} \cdot \vec{f}(\mathbf{x}, \mathbf{y})}}{Z_{\vec{\theta}}(\mathbf{x})} \tag{1}$$

where $\vec{f}$ is a feature vector function on parsed sentences and $\vec{\theta} \in \mathbb{R}^m$ parameterizes the model. Following McDonald et al. (2005a), we assume that the features are **edge-factored**:

$$\vec{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \vec{f}(\mathbf{x}, x_i, x_{\mathbf{y}(i)}) \tag{2}$$

In other words, the dependencies between words in the tree are all conditionally independent of each other, given the sequence $\mathbf{x}$ and the fact that the parse is a spanning tree. Despite the constraints they impose on features, edge-factored models have the advantage of tractable $O(n^3)$ inference algorithms or, with some trickery, $O(n^2)$ maximum *a posteriori* ("best parse tree") inference algorithms in the nonprojective case. Exact nonprojective inference and estimation become intractable if we break edge factoring (McDonald and Pereira, 2006).

We wish to estimate the parameters $\vec{\theta}$ by maximizing the conditional likelihood (like a CRF) rather

[1] To be precise, every word has in-degree 1, with the sole edge pointing from the word's parent, $x_{\mathbf{y}(i)} \rightarrow x_i$. x_0 has in-degree 0. By definition, trees are acyclic. The edges need not be planar and may "cross" in the plane, since we do not have a projectivity constraint. In some formulations, exactly one node in $\mathbf{x}$ can attach to x_0; here we allow multiple nodes to attach to x_0, since this occurs with some frequency in many existing datasets. Summation over trees where x_0 has exactly one child is addressed directly by Koo et al. (2007).

than the margin (McDonald et al., 2005a). For an empirical distribution $\tilde{p}$ given by a set of training examples, this means:

$$\max_{\vec{\theta}} \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x},\mathbf{y}) \left(\vec{\theta} \cdot \vec{f}(\mathbf{x},\mathbf{y})\right) - \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \log Z_{\vec{\theta}}(\mathbf{x}) \tag{3}$$

This optimization problem is typically solved using a quasi-Newton numerical optimization method such as L-BFGS (Liu and Nocedal, 1989). Such a method requires the gradient of the objective function, which for θ_k is given by the following difference in expectations of the value of feature f_k:

$$\frac{\partial}{\partial \theta_k} = \mathbf{E}_{\tilde{p}(\mathbf{X},\mathbf{Y})}\left[f_k(\mathbf{X},\mathbf{Y})\right] - \mathbf{E}_{\tilde{p}(\mathbf{X})p_{\vec{\theta}}(\mathbf{Y}|\mathbf{X})}\left[f_k(\mathbf{X},\mathbf{Y})\right] \tag{4}$$

The computation of $Z_{\vec{\theta}}(\mathbf{x})$ and the sufficient statistics (second expectation in Equation 4) are typically the difficult parts. They require summing the scores of all the spanning trees for a given sentence. Note that, in large-margin training, and in standard maximum *a posteriori* decoding, only a *maximum* over spanning trees is called for—it is *conditional training* that requires $Z_{\vec{\theta}}(\mathbf{x})$. In Section 3, we will show how this can be done exactly in $O(n^3)$ time.

3 Exploiting the Matrix Tree Theorem for $Z_{\vec{\theta}}(\mathbf{x})$

We wish to apply conditional training to estimate conditional models of nonprojective trees. This requires computing $Z_{\vec{\theta}}(\mathbf{x})$ for each training example (as an inner loop to training). In this section we show how the summation can be computed and how conditional training performs.

3.1 Kirchoff Matrix

Recall that we defined the unnormalized probability (henceforth, **score**) of a dependency tree as a combination of edge-factored scores for the edges present in the tree (Eq. 2):

$$\exp \vec{\theta}\cdot\vec{f}(\mathbf{x},\mathbf{y}) = \prod_{i=1}^{n} e^{\vec{\theta}\cdot\vec{f}(\mathbf{x},x_i,x_{\mathbf{y}(i)})} = \prod_{i=1}^{n} s_{\mathbf{x},\vec{\theta}}(i,\mathbf{y}(i)) \tag{5}$$

where $\mathbf{y}(i)$ denotes the parent of x_i in $\mathbf{y}$. $s_{\mathbf{x},\vec{\theta}}(i,j)$, then, denotes the (multiplicative) contribution of the edge from child i to parent j to the total score of the tree, if the edge is present. Define the **Kirchoff matrix** $\mathbf{K}_{\mathbf{x},\vec{\theta}} \in \mathbb{R}^{n\times n}$ by

$$\left[\mathbf{K}_{\mathbf{x},\vec{\theta}}\right]_{mom,kid} = \begin{cases} -s_{\mathbf{x},\vec{\theta}}(kid, mom) & \text{if } mom \neq kid \\ \sum_{j\in\{0,\ldots n\}:j\neq mom} s_{\mathbf{x},\vec{\theta}}(kid, j) & \text{if } mom = kid. \end{cases} \tag{6}$$

where mom indexes a parent node and kid a child node.

$\mathbf{K}_{\mathbf{x}\,\vec{\theta}}$ can be regarded as a special weighted adjacency matrix in which the ith diagonal entry is the sum of edge-scores directed into vertex i (i.e., x_i is the child)—note that the sum includes the score of attaching x_i to the wall x_0.

In our notation and in one specific form, the Matrix Tree Theorem (Tutte, 1984) states:[2]

Theorem 1 *The determinant of the Kirchoff matrix* $\mathbf{K}_{\mathbf{x},\vec{\theta}}$ *is equal to the sum of scores of all directed spanning trees in* $\mathcal{Y}_{\mathbf{x}}$ *rooted at* x_0. *Formally:*

$$\left|\mathbf{K}_{\mathbf{x},\vec{\theta}}\right| = Z_{\vec{\theta}}(\mathbf{x}).$$

A proof is omitted; see Tutte (1984).

To compute $Z_{\vec{\theta}}(\mathbf{x})$, we need only take the determinant of $\mathbf{K}_{\mathbf{x},\vec{\theta}}$, which can be done in $O(n^3)$ time using the standard LU factorization to compute the matrix inverse. Since all of the edge weights used to construct the Kirchoff matrix are positive, it is diagonally dominant and therefore non-singular (i.e., invertible).

3.2 Gradient

The gradient of $Z_{\vec{\theta}}(\mathbf{x})$ (required for numerical optimization; see Eqs. 3–4) can be efficiently computed from the same matrix inverse. While $\nabla \log Z_{\vec{\theta}}(\mathbf{x})$ equates to a vector of feature expectations (Eq. 4), we exploit instead some facts from linear algebra

[2]There are proven generalizations of this theorem (Chen, 1965; Chaiken, 1982; Minoux, 1999); we give the most specific form that applies to our case, originally proved by Tutte in 1948. Strictly speaking, our $\mathbf{K}_{\mathbf{x},\vec{\theta}}$ is not the Kirchoff matrix, but rather a *submatrix* of the Kirchoff matrix with a leftmost column of zeroes and a topmost row $[0, -s_{\mathbf{x},\vec{\theta}}(1,0), ..., -s_{\mathbf{x},\vec{\theta}}(n,0)]$ removed. Farther afield, Jaakkola et al. (1999) used an undirected matrix tree theorem for learning tree structures for graphical models.

$$\mathbf{K}_{\mathbf{x},\vec{\theta}} = \begin{bmatrix} \sum_{j \in \{0,\ldots,n\}: j \neq 1} s_{\mathbf{x},\vec{\theta}}(1,j) & -s_{\mathbf{x},\vec{\theta}}(2,1) & \cdots & -s_{\mathbf{x},\vec{\theta}}(n,1) \\ -s_{\mathbf{x},\vec{\theta}}(1,2) & \sum_{j \in \{0,\ldots,n\}: j \neq 2} s_{\mathbf{x},\vec{\theta}}(2,j) & \cdots & -s_{\mathbf{x},\vec{\theta}}(n,2) \\ \vdots & \vdots & \ddots & \vdots \\ -s_{\mathbf{x},\vec{\theta}}(1,n) & -s_{\mathbf{x},\vec{\theta}}(2,n) & \cdots & \sum_{j \in \{0,\ldots,n\}: j \neq n} s_{\mathbf{x},\vec{\theta}}(n,j) \end{bmatrix}$$

and the chain rule. First, note that, for any weight θ_k,

$$\begin{aligned} &\frac{\partial \log Z_{\vec{\theta}}(\mathbf{x})}{\partial \theta_k} \\ &= \frac{\partial \log |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \sum_{i=1}^{n} \sum_{j=0}^{n} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \frac{\partial s_{\mathbf{x},\vec{\theta}}(i,j)}{\partial \theta_k} \\ &= \frac{1}{|\mathbf{K}_{\mathbf{x},\vec{\theta}}|} \sum_{i=1}^{n} \sum_{j=0}^{n} s_{\mathbf{x},\vec{\theta}}(i,j) f_k(\mathbf{x}, x_i, x_j) \\ &\qquad \times \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \end{aligned} \tag{7}$$

(We assume $s_{\mathbf{x},\vec{\theta}}(i,i) = 0$, for simplicity of notation.) The last line follows from the definition of $s_{\mathbf{x},\vec{\theta}}(i,j)$ as $\exp \vec{\theta} \cdot \vec{f}(\mathbf{x}, x_i, x_j)$. Now, since $s_{\mathbf{x},\vec{\theta}}(i,j)$ affects the Kirchoff matrix in *at most* two cells—(i,i) and (j,i), the latter only when $j > 0$—we know that

$$\begin{aligned} \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} &= \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}} \frac{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}}{\partial s_{\mathbf{x},\vec{\theta}}(i,i)} \\ &\quad - \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}} \frac{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}}{\partial s_{\mathbf{x},\vec{\theta}}(i,j)} \\ &= \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{i,i}} - \frac{\partial |\mathbf{K}_{\mathbf{x},\vec{\theta}}|}{\partial [\mathbf{K}_{\mathbf{x},\vec{\theta}}]_{j,i}} \end{aligned} \tag{8}$$

We have now reduced the problem of the gradient to a linear function of $\nabla |\mathbf{K}_{\mathbf{x},\vec{\theta}}|$ with respect to the cells of the matrix itself. At this point, we simplify notation and consider an arbitrary matrix $\mathbf{A}$.

The minor $m_{j,i}$ of a matrix $\mathbf{A}$ is the determinant of the submatrix obtained by striking out row j and column i of $\mathbf{A}$; the cofactor $c_{j,i}$ of $\mathbf{A}$ is then $(-1)^{i+j} m_{j,i}$. Laplace's formula defines the determinant as a linear combination of matrix cofactors of an arbitrary row j:

$$|\mathbf{A}| = \sum_{i=1}^{n} [\mathbf{A}]_{j,i} c_{j,i} \tag{9}$$

It should be clear that any $c_{j,k}$ is constant with respect to the cell $[\mathbf{A}]_{j,i}$ (since it is formed by removing row j of $\mathbf{A}$) and that other entries of $\mathbf{A}$ are constant with respect to the cell $[\mathbf{A}]_{j,i}$. Therefore:

$$\frac{\partial |\mathbf{A}|}{\partial [\mathbf{A}]_{j,i}} = c_{j,i} \tag{10}$$

The inverse matrix $\mathbf{A}^{-1}$ can also be defined in terms of cofactors:

$$[\mathbf{A}^{-1}]_{i,j} = \frac{c_{j,i}}{|\mathbf{A}|} \tag{11}$$

Combining Eqs. 10 and 11, we have:

$$\frac{\partial |\mathbf{A}|}{\partial [\mathbf{A}]_{j,i}} = |\mathbf{A}| [\mathbf{A}^{-1}]_{i,j} \tag{12}$$

Plugging back in through Eq. 8 to Eq. 7, we have:

$$\begin{aligned} \frac{\partial \log Z_{\vec{\theta}}(\mathbf{x})}{\partial \theta_k} &= \sum_{i=1}^{n} \sum_{j=0}^{n} s_{\mathbf{x},\vec{\theta}}(i,j) f_k(\mathbf{x}, x_i, x_j) \\ &\quad \times \left(\left[\mathbf{K}_{\mathbf{x},\vec{\theta}}^{-1} \right]_{i,i} - \left[\mathbf{K}_{\mathbf{x},\vec{\theta}}^{-1} \right]_{i,j} \right) \end{aligned} \tag{13}$$

where $[\mathbf{K}^{-1}]_{i,0}$ is taken to be 0. Note that the cofactors do not need to be computed directly. We proposed in Section 3.1 to get $Z_{\vec{\theta}}(\mathbf{x})$ by computing the inverse of the Kirchoff matrix (which is known to exist). Under that procedure, the marginalization is a by-product of the gradient.

decode	train	Arabic	Czech	Danish	Dutch	
m*ap*	MIRA	79.9	**81.4**	86.6	**90.0**	
	CE	80.4	80.2	**87.5**	**90.0**	(Section 3)
mBr	MIRA	79.4	80.3	85.0	87.2	(Section 4)
	CE	**80.5**	80.4	**87.5**	**90.0**	(Sections 3 & 4)

Table 1: Unlabeled dependency parsing accuracy (on test data) for two training methods (MIRA, as in McDonald et al. (2005b), and conditional estimation) and with maximum *a posteriori* (m*ap*) and minimum Bayes-risk (mBr) decoding. **Boldface** scores are best in their column on a permutation test at the .05 level.

3.3 Experiment

We compare conditional training of a nonprojective edge-factored parsing model to the online MIRA training used by McDonald et al. (2005b). Four languages with relatively common nonprojective phenomena were tested: Arabic (Hajič et al., 2004), Czech (Böhmová et al., 2003), Danish (Kromann, 2003), and Dutch (van der Beek et al., 2002). The Danish and Dutch datasets were prepared for the CoNLL 2006 shared task (Buchholz and Marsi, 2006); Arabic and Czech are from the 2007 shared task. We used the same features, extracted by McDonald's code, in both MIRA and conditional training. In this paper, we consider only *unlabeled* dependency parsing.

Our conditional training used an online gradient-based method known as stochastic gradient descent (see, e.g., Bottou, 2003). Training with MIRA and conditional estimation take about the same amount of time: approximately 50 sentences per second. Training proceeded as long as an improvement on held-out data was evident. The accuracy of the hypothesized parses for the two models, on each language, are shown in the top two rows of Tab. 1 (labeled "m*ap*" for maximum *a posteriori*, meaning that the highest-weighted tree is hypothesized).

The two methods are, not surprisingly, close in performance; conditional likelihood outperformed MIRA on Arabic and Danish, underperformed MIRA on Czech, and the two tied on Dutch. Results are significant at the .05 level on a permutation test. Conditional estimation is in practice more prone to over-fitting than maximum margin methods, though we did not see any improvement using zero-mean Gaussian priors (variance 1 or 10).

These experiments serve to validate conditional estimation as a competitive learning algorithm for parsing models, and the key contribution of the summing algorithm that permits conditional estimation.

4 Minimum Bayes-Risk Decoding

A second application of probability distributions over trees is the alternative decoding algorithm known as **minimum Bayes-risk** (mBr) decoding. The more commonly used maximum *a posteriori* decoding (also known as "Viterbi" decoding) that we applied in Section 3.3 sought to minimize the expected whole-tree loss:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} p_{\vec{\theta}}(\mathbf{y} \mid \mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[-\delta(\mathbf{y}, \mathbf{Y})\right] \tag{14}$$

Minimum Bayes-risk decoding generalizes this idea to an arbitrary loss function ℓ on the proposed tree:

$$\hat{\mathbf{y}} = \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[\ell(\mathbf{y}, \mathbf{Y})\right] \tag{15}$$

This technique was originally applied in speech recognition (Goel and Byrne, 2000) and translation (Kumar and Byrne, 2004); Goodman (1996) proposed a similar idea in probabilistic context-free parsing, seeking to maximize expected recall. For more applications in parsing, see Petrov and Klein (2007).

The most common loss function used to evaluate dependency parsers is the number of attachment errors, so we seek to decode using:

$$\begin{aligned}\hat{\mathbf{y}} &= \operatorname*{argmin}_{\mathbf{y}} \mathbf{E}_{p_{\vec{\theta}}(\mathbf{Y}|\mathbf{x})}\left[\sum_{i=1}^{n} -\delta(\mathbf{y}(i), \mathbf{Y}(i))\right] \\ &= \operatorname*{argmax}_{\mathbf{y}} \sum_{i=1}^{n} p_{\vec{\theta}}(\mathbf{Y}(i) = \mathbf{y}(i) \mid \mathbf{x}) \end{aligned} \tag{16}$$

To apply this decoding method, we make use of Eq. 13, which gives us the posterior probabilities of edges under the model, and the same Chiu-Liu-Edmonds maximum directed spanning tree algorithm used for maximum *a posteriori* decoding. Note that this decoding method can be applied regardless of how the model is trained. It merely requires assuming that the tree scores under the trained model (probabilistic or not) can be treated as unnormalized log-probabilities over trees given the sentence $\mathbf{x}$.

We applied minimum Bayes-risk decoding to the models trained using MIRA and using conditional estimation (see Section 3.3). Table 1 shows that, across languages, minimum Bayes-risk decoding *hurts* slightly the performance of a MIRA-trained model, but *helps* slightly or does not affect the performance of a conditionally-trained model. Since MIRA does not attempt to model the distribution over trees, this result is not surprising; interpreting weights as defining a conditional log-linear distribution is questionable under MIRA's training criterion.

One option, which we do not test here, is to use minimum Bayes-risk decoding *inside* of MIRA training, to propose a hypothesis tree (or k-best trees) at each *training* step. Doing this would more closely match the training conditions with the testing conditions; however, it is unclear whether there is a formal interpretation of such a combination, for example its relationship to McDonald et al.'s "factored MIRA."

Minimum Bayes-risk decoding, we believe, will become important in nonprojective parsing with *non-edge-factored* models. Note that minimium Bayes-risk decoding reduces *any* parsing problem to the maximum directed spanning tree problem, even if the original model is not edge-factored. All that is required are the marginals $p_{\vec{\theta}}(\mathbf{Y}(i) = \mathbf{y}(i) \mid \mathbf{x})$, which may be intractable to compute exactly, though it may be possible to develop efficient approximations.

5 Hidden Variables

A third application of probability distributions over trees is hidden-variable learning. The Expectation-Maximization (EM) algorithm (Baum and Petrie, 1966; Dempster et al., 1977; Baker, 1979), for example, is a way to maximum the likelihood of training data, marginalizing out hidden variables. This has been applied widely in unsupervised parsing (Carroll and Charniak, 1992; Klein and Manning, 2002). More recently, EM has been used to learn hidden variables in parse trees; these can be head-child annotations (Chiang and Bikel, 2002), latent head features (Matsuzaki et al., 2005; Prescher, 2005; Dreyer and Eisner, 2006), or hierarchically-split nonterminal states (Petrov et al., 2006).

To date, we know of no attempts to apply hidden variables to supervised dependency tree models. If the trees are constrained to be projective, EM is easily applied using the inside-outside variant of the parsing algorithm described by Eisner (1996) to compute the marginal probability. Moving to the nonprojective case, there are two difficulties: (a) we must marginalize over nonprojective trees and (b) we must define a generative model over $(\mathbf{X}, \mathbf{Y})$.

We have already shown in Section 3 how to solve (a); here we avoid (b) by maximizing *conditional* likelihood, marginalizing out the hidden variable, denoted $\mathbf{z}$:

$$\max_{\vec{\theta}} \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}, \mathbf{y}) \log \sum_{\mathbf{z}} p_{\vec{\theta}}(\mathbf{y}, \mathbf{z} \mid \mathbf{x}) \tag{17}$$

This sort of conditional training with hidden variables was carried out by Koo and Collins (2005), for example, in reranking; it is related to the information bottleneck method (Tishby et al., 1999) and contrastive estimation (Smith and Eisner, 2005).

5.1 Latent Dependency Labels

Noting that our model is edge-factored (Eq. 2), we define our hidden variables to be edge-factored as well. We can think of the hidden variables as clusters on dependency tokens, and redefine the score of an edge to be:

$$s_{\mathbf{x},\vec{\theta}}(i, j) = \sum_{z \in \mathcal{Z}} e^{\vec{\theta} \cdot \vec{f}(\mathbf{x}, x_i, x_j, z)} \tag{18}$$

where $\mathcal{Z}$ is a set of dependency clusters.

Note that keeping the model edge-factored means that the cluster of each dependency in a tree is conditionally independent of all the others, given the words. This is computationally advantageous (we can factor out the marginalization of the hidden variable by edge), and it permits the use of any clustering method at all. For example, if an auxiliary clustering model $q(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$—perhaps one that did *not* make such independence assumptions—were used, the posterior probability $q(Z_i = z \mid \mathbf{x}, \mathbf{y})$ could be a feature in the proposed model. On the other hand, we must consider carefully the role of the dependency clusters in the model; if clusters are learned extrinsic to estimation of the parsing model, we should not expect them to be directly advantageous to parsing accuracy.

5.2 Experiments

We tried two sets of experiments with clustering. In one case, we simply augmented all of McDonald et al.'s edge features with a cluster label in hopes of improved accuracy. Models were initialized near zero, with Gaussian noise added to break symmetry among clusters.

Under these conditions, performance stayed the same or changed slightly (see Table 2); none of the improvements are significant. Note that three decoders were applied: maximum *a posteriori* (m*ap*) and minimum Bayes-risk (mBr) as described in Section 4, and "max-z," in which each possible edge was labeled and weighted only with its most likely cluster (rather than the sum over all clusters), before finding the most probable tree.[3] For each of the three languages tested, some number of clusters and some decoding method gave small improvements over the baseline.

More ambitiously, we hypothesized that many lexicalized features on edges could be "squeezed" through clusters to reduce the size of the feature set. We thus removed all word-word and lemma-lemma features and all tag fourgrams. Although this reduced our feature set by a factor of 60 or more (prior to taking a cross-product with the clusters), the damage of breaking the features was tremendous, and performance even with a thousand clusters barely broke 25% accuracy.

6 Discussion

Noting that adding latent features to nonterminals in unlexicalized context-free parsing has been very successful (Chiang and Bikel, 2002; Matsuzaki et al., 2005; Prescher, 2005; Dreyer and Eisner, 2006; Petrov et al., 2006), we were surprised not to see a more substantial performance improvement through latent features. We propose several interpretations. First, it may simply be that many more clusters may be required. Note that the label-set sizes for the labeled versions of these datasets are larger than 32 (e.g., 50 for Danish). This has the unfortunate effect of blowing up the feature space beyond the memory capacity of our machines (hence our attempts at squeezing high-dimensional features through the clusters).

Of course, improved clustering methods may also improve performance. In particular, a cluster-learning algorithm that permits clusters to split and/or merge, as in Petrov et al. (2006) or in Pereira et al. (1993), may be appropriate.

Given the relative simplicity of clustering methods for context-free parsing to date (gains were found just by using Expectation-Maximization), we believe the fundamental reason clustering was not particularly helpful here is a structural one. In context-free parsing, the latent features are (in published work to date) on nonterminal states, which are the stuctural "bridge" between context-free rules. Adding features to those states is a way of pushing information—encoded indirectly, perhaps—farther around the tree, and therefore circumventing the strict independence assumptions of probabilistic CFGs.

In an edge-factored dependency model, on the

# cl.	decoding	Arabic	Danish	Dutch
none	m*ap*=max-z	80.4	87.5	90.0
	mBr	80.5	87.5	90.0
2	m*ap*	80.4	87.5	89.5
	mBr	80.6	87.3	89.7
	max-z	80.4	86.3	89.4
16	m*ap*	80.4	87.6	90.1
	mBr	80.4	87.6	90.1
	max-z	80.4	87.6	90.2
32	m*ap*	80.0	87.6	–
	mBr	80.4	87.5	–
	max-z	80.0	87.5	–

Table 2: Augmenting edge features with clusters results in similar performance to conditional training with no clusters (top two lines). Scores are unlabeled dependency accuracy on test data.

[3]Czech experiments were not done, since the number of features (more than 14 million) was too high to multiply out by clusters.

other hand, latent features on the edges seem to have little effect. Given that they are locally "summed out" when we compute the scores of possible attachments, it should be clear that the edge clusters do not circumvent any independence assumptions. Three options appear to present themselves. First, we might attempt to learn clusters in tandem with estimating a richer, non-edge-factored model which would require approximations to $Z_{\vec{\theta}}(\mathbf{x})$, if conditional training were to be used. Note that the approximations to maximizing over spanning trees with second-order features, proposed by McDonald and Pereira (2006), do not permit estimating the clusters as part of the same process as weight estimation (at least not without modification). In the conditional estimation case, a variational approach might be appropriate. The second option is to learn clusters offline, *before* estimating the parser. (We suggested how to incorporate soft clusters into our model in Section 5.1.) This option is computationally advantageous but loses sight of the aim of learning the clusters specifically to improve parsing accuracy. Third, noting that the structural "bridge" between two coincident edges is the shared vertex (word), we might consider *word* token clustering.

We also believe this structural locality issue helps explain the modesty of the gains using minimum Bayes-risk decoding with conditional training (Section 4). In other dependency parsing scenarios, minimum Bayes-risk decoding has been found to offer significant advantages—why not here? Minimum Bayes-risk makes use of global statistical dependencies in the posterior when making local decisions. But in an edge-factored model, the edges are all conditionally independent, given that $\mathbf{y}$ is a spanning tree.

As a *post hoc* experiment, we compared purely greedy attachment (attach each word to its maximum-weighted parent, without any tree constraints). Edge scores as defined in the model were compared to minimum Bayes-risk posterior scores, and the latter were consistently better (though this always under-performed optimal spanning-tree decoding, unsurprisingly). This comparison serves only to confirm that minimum Bayes-risk decoding is a way to circumvent independence assumptions (here made by a decoder), but only when the trained model *does not* make those particular assumptions.

7 Conclusion

We have shown how to carry out exact marginalization under an edge-factored, conditional log-linear model over nonprojective dependency trees. The method has cubic runtime in the length of the sequence, but is very fast in practice. It can be used in conditional training of such a model, in minimum Bayes-risk decoding (regardless of how the model is trained), and in training with hidden variables. We demonstrated how each of these techniques gives results competitive with state-of-the-art existing dependency parsers.

Acknowledgments

The authors thank the anonymous reviewers, Jason Eisner, Keith Hall, and Sanjeev Khudanpur for helpful comments, and Michael Collins and Ryan McDonald for sharing drafts of their related, concurrent papers. This work was supported in part by NSF ITR grant IIS-0313193.

References

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Proc. of the Acoustical Society of America*, pages 547–550.

P. Bartlett, M. Collins, B. Taskar, and D. McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in NIPS 17*.

L. E. Baum and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer.

L. Bottou. 2003. Stochastic learning. In *Advanced Lectures in Machine Learning*, pages 146–168. Springer.

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

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. Technical report, Brown University.

S. Chaiken. 1982. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic and Discrete Methods*, 3(3):319–329.

W.-K. Chen. 1965. Topological analysis for active networks. *IEEE Transactions on Circuit Theory*, 12(1):85–91.

D. Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *Proc. of COLING*.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. of ACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

M. Dreyer and J. Eisner. 2006. Better informed training of latent syntactic features. In *Proc. of EMNLP*.

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

V. Goel and W. Byrne. 2000. Minimum Bayes risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

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*.

J. Hall, J. Nivre, and J. Nilsson. 2006. Discriminative learning for data-driven dependency parsing. In *Proc. of COLING-ACL*.

T. Jaakkola, M. Meila, and T. Jebara. 1999. Maximum entropy discrimination. In *Advances in NIPS 12*.

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

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*.

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*.

T. Koo, A. Globerson, X. Carreras, and M. Collins. 2007. Structured prediction models via the Matrix-Tree Theorem. In *Proc. of EMNLP-CoNLL*.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT*.

S. Kumar and W. Byrne. 2004. Minimum Bayes risk decoding for statistical machine translation. In *Proc. of HLT-NAACL*.

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

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of ACL*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*.

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

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

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

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

M. Minoux. 1999. A generalization of the all minors matrix tree theorem to semirings. *Discrete Mathematics*, 199:139–150.

Y. Miyao and J. Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT*.

F. C. N. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proc. of the 31st ACL*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. of COLING-ACL*.

D. Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *Proc. of IWPT*.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*.

A. Ratnaparkhi, S. Roukos, and R. T. Ward. 1994. A maximum entropy model for parsing. In *Proc. of ICSLP*.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL*.

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

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

N. Tishby, F. C. N. Pereira, and W. Bialek. 1999. The information bottleneck method. In *Proc. of the 37th Allerton Conference on Communication, Control and Computing*, pages 368–377.

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

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *CLIN*.