Posterior Sparsity in Unsupervised Dependency Parsing

Jennifer Gillenwater[*] Kuzman Ganchev[†] João Graça[‡]
Fernando Pereira[**] Ben Taskar[††]

[*]University of Pennsylvania
[†]University of Pennsylvania
[‡]L^{2}F INESC-ID
[**]Google Inc.
[††]University of Pennsylvania, taskar@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-19.

Posterior Sparsity in Unsupervised Dependency Parsing

Jennifer Gillenwater jengi@cis.upenn.edu
Kuzman Ganchev kuzman@cis.upenn.edu
Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA

João Graça joao.graca@l2f.inesc-id.pt
L^2F INESC-ID
Lisboa, Portugal

Fernando Pereira pereira@google.com
Google Inc.
Mountain View, CA, USA

Ben Taskar taskar@cis.upenn.edu
Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA

Abstract

A strong inductive bias is essential in unsupervised grammar induction. In this paper, we explore a particular sparsity bias in dependency grammars that encourages a small number of unique dependency types. We use part-of-speech (POS) tags to group dependencies by parent-child types and investigate sparsity-inducing penalties on the posterior distributions of parent-child POS tag pairs in the posterior regularization (PR) framework of Graça et al. (2007). In experiments with 12 different languages, we achieve significant gains in directed accuracy over the standard expectation maximization (EM) baseline for 9 of the languages, with an average accuracy improvement of 6%. Further, we show that for 8 out of 12 languages, the new method outperforms models based on standard Bayesian sparsity-inducing parameter priors, with an average improvement of 4%. On English text in particular, we show that our approach improves performance over other state of the art techniques.

1. Introduction

We investigate unsupervised learning methods for dependency parsing models that impose sparsity biases on the types of dependencies. We assume a corpus annotated with part-of-speech (POS) tags, where the task is to induce a dependency model from the tag sequences for corpus sentences. In this setting, the *type* of a dependency is defined as a simple pair: tag of the dependent (also known as the child), and tag of the head (also known as the parent) for that dependent. Given that POS tags are typically designed to convey information about grammatical relations, it is reasonable to assume that only some of the possible dependency types would be realized for any given language. For instance, it is ungrammatical for nouns to dominate verbs, adjectives to dominate adverbs, and determiners to dominate almost any part of speech. In other words, the realized dependency types should be a sparse subset of all the possible types.

Previous work in unsupervised grammar induction has tried to achieve sparsity through priors on model parameters. For instance, Liang et al. (2007), Finkel et al. (2007) and Johnson et al. (2007) experimented with hierarchical Dirichlet process priors and Headden III et al. (2009) proposed a discounting Dirichlet prior. Such priors on parameters encourage a standard generative dependency parsing model (see Section 2) to limit the number of dependent types for each head type. Although not focused on sparsity, several other studies use soft parameter sharing to constrain the capacity of the model and hence couple different types of dependencies. To this end, Cohen et al. (2008) and Cohen and Smith (2009) investigatde a (shared) logistic normal prior, and Headden III et al. (2009) used a backoff scheme.

Our experiments (see Section 5) show that the more effective sparsity pattern is one that limits the total number of unique head-dependent tag pairs. Unlike sparsity-inducing parameter priors, this kind of sparsity bias does not induce competition between dependent types for each head type. As we show in Section 4, we can achieve the desired bias with a sparsity constraint on model posteriors, using the posterior regularization (PR) framework (Graça et al., 2007).

Specifically, to implement PR we augment the maximum likelihood objective of the generative dependency model with a term that penalizes head tag-dependent tag distributions that are too permissive. We consider two choices for the form of the penalty, and show experimentally that the following penalty works especially well: the model pays for the first time it selects a word with tag c as a dependent of a head with tag p; after that, choosing a head with that p for any other occurrence of c is free.

The remainder of this paper is organized as follows. Section 2 reviews the generative model for dependency parsing. Section 3 summarizes previous approaches to learning with this model, focusing in particular on an attempt to induce parameter sparsity via a parameter prior. Section 4 describes learning with PR constraints and how to encode posterior sparsity under the PR framework. Section 5 describes the results of dependency parsing experiments across 12 languages and against recent published state of art results for the English language. Section 6 analyzes these results, explaining why PR manages to learn where other methods fail, and Section 7 concludes. The model and all the code required to reproduce the experiments will be available online at `code.google.com/p/pr-toolkit` soon.

2. Parsing Model

The models we consider are based on Klein and Manning (2004)'s dependency model with valence (DMV). We also investigate extensions to the DMV borrowed from McClosky (2008) and Headden III et al. (2009). These extensions are not crucial to our experimental success with posterior regularization, but we choose to explore them for better comparison with previous work. As will be discussed in the experiments section, both for the basic and for the extended models accuracy can be increased by applying posterior regularization. Section 2.1 describes the basic model and Section 2.2 describes the extensions we implemented.

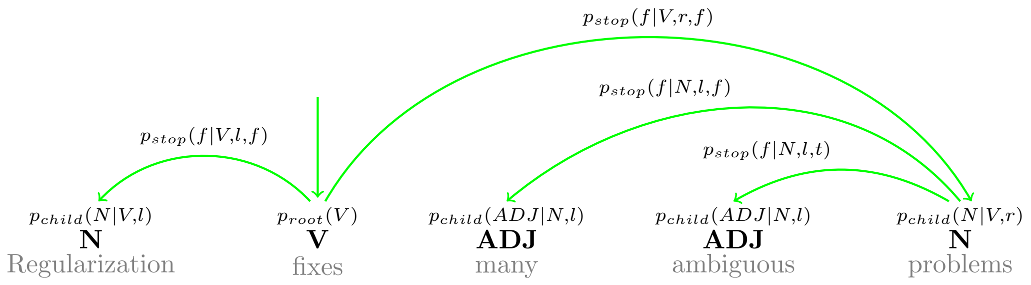


Figure 1: Example of a dependency tree with DMV probabilities. Right-dependents of a head are denoted by r, left-dependents by l. The letters t and f denote 'true' and 'false.' For example, in $p_{stop}(f \mid V, r, f)$ the f to the left of the conditioning bar indicates that the model has decided *not* to stop, and the other f indicates V does *not* yet have any right dependents. Note that the $p_{stop}(t \mid \ldots)$ are ommitted in this diagram.

2.1 Dependency Model With Valence (DMV)

The DMV model specifies the following generative process. For a sentence consisting of POS tags $\mathbf{x}$, the root head POS $r(\mathbf{x})$ is generated first with probability $p_{root}(r(\mathbf{x}))$. For example, in Figure 1 this corresponds to generating the V with probability $p_{root}(V)$.

After generating the root, the model next generates dependents of the root. First, it generates right dependents. It decides whether to produce a right dependent conditioned on the identity of the root and the fact that it currently has no other right dependents. In our example, this decision is represented by the probability $p_{stop}(f \mid V, r, f)$. If it decides to generate a right dependent, it generates a particular dependent POS by conditioning on the fact that the head POS is $r(\mathbf{x})$ and that the directionality is to the right. In our example, this corresponds to the probability $p_{child}(N \mid V, r)$. The model then returns to the choice of whether or not to stop generating right dependents, this time conditioned on the fact that it already has at least right dependent. In our example, this corresponds to the probability $p_{stop}(t \mid V, r, t)$, which indicates that the model is done generating right dependents of V.

After stopping the generation of right dependents, the model generates left-dependents using the mirror reversal of the previous process. Once the root has generated all of its dependents, the dependents generate their own dependents in the same manner.

You may note that in Figure 1 the leftmost dependent of the final N is generated before the other left dependent. The convention we are using here is that the model generates dependents starting with the rightmost one, moving inward (leftward) until all right dependents are added, then it generates the leftmost left dependent and moves inward (rightward) from there. This convention has no effect on the final probability of a parse tree under the basic DMV. However, as we will note in the following subsection, it does affect dependency tree probabilities in the extended model.

2.2 Model Extensions

We implemented three model extensions, borrowed from McClosky (2008) and Headden III et al. (2009). The first extension relates to the stop probabilities, and the second two relate to dependent probabilities.

2.2.1 Extending Stop Probabilities

This extension conditions whether to stop generating dependents in a given direction on a larger set of previous decisions. Specifically, the probability of stopping in a particular direction depends not only on whether there are any dependents in that direction already, but also on how many. In the example of Figure 1, this corresponds to changing $p_{stop}(f \mid V, r, f)$ to $p_{stop}(f \mid V, r, 0)$ and similarly for all the other stop probabilities. The 0 in this case indicates that V has no other right dependents when it decides whether to continue generating right dependents.

In later sections of this paper, when we talk about a model with maximum stop valency S, this means we distinguish the cases of $0, 1, \ldots, S-1$, and $\geq S$ dependents in a given direction. The basic DMV has maximum stop valency 1 because it distinguishes between having zero dependents and at least one dependent in a given direction. A model with maximum stop valency of 2 would distinguish between having 0, 1, or at least 2 dependents in a particular direction. In this case, when a head generates more dependents in a particular direction after its second dependent, the stopping distribution it draws from will always be the same—for head p and direction d this will be $p_{stop}(\cdot \mid p, d, 2)$.

With this model extension, the order in which dependents are generated becomes relevant to the probability of an overall parse tree. We choose to stick with the conventional generation order described in Section 2.1. In cases where the identity of the rightmost and leftmost dependents have a greater influence on the true stop probability than the inner dependents, this ordering will work to the model's advantage. We do not investigate in this work which languages this holds true for, though changing this ordering might be one additional way to increase parsing accuracy for some languages.

2.2.2 Extending Dependent Probabilities

The second model extension we implement is analogous to the first, but applies to dependent tag probabilities instead of stop probabilities. That is, we expand the set of variables the model conditions on when selecting a particular dependent tag. Again, what condition on is how many other dependents were already generated in the same direction. For the example in Figure 1, this means $p_{child}(N \mid V, r)$ becomes $p_{child}(N \mid V, r, 0)$ and similarly for all other p_{child}. In later sections of this paper, when we talk about a model with maximum child valency C, this means we distinguish between having $0, 1, \ldots, C-1$, and $\geq C$ dependents in a particular direction. The basic DMV has maximum child valency 0 because it does not make these distinctions.

This extension to the child probabilities dramatically increases model complexity. Specifically, the number of parameters grows as $O(CT^2)$. Thus, the third and final model extension we implement is to add a backoff for the child probabilities that does not condition on the identity of the parent POS (see Equation 2).

2.2.3 Complete Model

Formally, under the extended DMV the probability of a sentence with POS tags $\mathbf{x}$ and dependency tree $\mathbf{y}$ is given by:

$$\begin{aligned}
&p_\theta(\mathbf{x}, \mathbf{y}) = p_{root}(r(\mathbf{x}))\times \\
&\prod_{y \in \mathbf{y}} p_{stop}(false \mid y_p, y_d, y_{v_s}) p_{child}(y_c \mid y_p, y_d, y_{v_c})\times \\
&\prod_{x \in \mathbf{x}} p_{stop}(true \mid x, left, x_{v_l})\, p_{stop}(true \mid x, right, x_{v_r})
\end{aligned} \tag{1}$$

where $r(\mathbf{x})$ is the root tag of the dependency tree, y is the dependency of y_c on head y_p in direction y_d, and y_{v_c}, y_{v_s}, x_{v_r}, and x_{v_l} indicate valence. To formally define these last four variables, first let V_c denote the model's maximum child valency and let V_s denote maximum stop valency. Further, let a_{cpd} to be the number of y_p's dependents that are further in direction y_d than y_c, and a_{xl} (a_{xr}) be the total number of dependents of parent x to the left (right). Then we can formally express the valency variables as:

$$\begin{aligned}
y_{v_c} &= \min(V_c, a_{cpd}), & y_{v_s} &= \min(V_s, a_{cpd}) \\
x_{v_l} &= \min(V_s, a_{xl}), & x_{v_r} &= \min(V_s, a_{xr}).
\end{aligned}$$

In the third model extension, the backoff for the child probability to a probability not dependent on parent POS, $p_{child}(y_c \mid y_d, y_{v_c})$, can formally be expressed by:

$$\lambda p_{child}(y_c \mid y_p, y_d, y_{v_c}) + (1-\lambda) p_{child}(y_c \mid y_d, y_{v_c}) \tag{2}$$

for $\lambda \in [0, 1]$. In Headden III et al. (2009) λ is a learned model parameter. In our experiments, we do not try to tune λ, but rather fix it at $1/3$. This is a crude approximation to the value used by Headden III et al. (2009) (see Section 3.2.2 for more details).

2.3 Model Initialization

DMV model parameter initialization plays a crucial role in the learned model's accuracy because of local maxima in the likelihood function. Klein and Manning (2004) use an "harmonic initializer", which we will refer on this paper as K&M. This initialization uses the posteriors for a fake E-step as initial parameter: posterior root probabilities are uniform $p_{root}(r(\mathbf{x})) = \frac{1}{|\mathbf{x}|}$ and head-dependent probabilities are inversely proportional to the string distance between head and dependent, $p_{child}(y_c \mid y_p, y_d, y_{v_c}) \propto \frac{1}{|y_p - y_c|}$, normalized to form a proper probability distribution. This initialization biases the parameters to prefer local attachments.

Smith (2006) compares K&M with random initialization and with uniform initialization. These two alternatives were worse than K&M unless some labeled data could be used to help pick a set of random parameters. Headden III et al. (2009) suggested using random pools to initialize the extended model described in Subsection 2.2 to avoid the very strong tie between K&M and the DMV. A random pool consists of a set of B randomly initialized models trained for a small number of iterations. From these B models, the one that assigns highest likelihood to held-out development data is picked and trained until convergence. M such pools are used to create M final models, whose mean accuracy and standard deviation are reported. We will refer to this initialization method as RandomP; it performs significantly better than K&M.

Recently, Spitkovsky et al. (2010) presented several initialization methods that aim to gradually increase the complexity of the model, as measured by the size of the search space, which in the DMV model is exponential in the size of the sentence length. The Baby Steps (BS) method starts by training the model in sentences of length 1 where there is no ambiguity but nevertheless some information about heads can be gleaned. The parameters of this model are used to initialize a training run over sentences of length 2, and so on, up to a maximum length. The second method, Less is More (LsM), uses information from the BS method to pick a sentence length that includes enough sentences to train a model with good predictive power, but leaves out longer sentences that do not add much information. The hybrid method Leapfrog (LP) combines the models from the two previous approaches. All of these methods improve over the K&M initialization. In this paper, we use K&M initialization for all experiments for simplicity. However , to achieve a fair comparison with related work, we report the accuracy differences from using different initialization methods in Subsection 5.3.

3. Previous Learning Approaches

The main comparisons for our sparse learning methods will be the expectation maximization (EM) method and Bayesian learning with a sparsity-inducing prior. We will also compare our accuracy to that achieved by several methods that use other priors. This latter comparison will be less direct though, as these priors tend to encode linguistic information at a finer-grained level. Before we make an empirical comparison in Section 5, in this section we review the theory behind the EM method and Bayesian learning methods. In what follows, we will denote the entire unlabeled corpus by $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ and a set of corresponding parses for each corpus sentence by $\mathbf{Y} = \{\mathbf{y}^1, \ldots, \mathbf{y}^n\}$.

3.1 Expectation Maximization

The EM algorithm is a popular method for optimizing marginal likelihood $\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} p_\theta(\mathbf{X}, \mathbf{Y})$. We briefly review here the interpretation of the EM algorithm given by Neal and Hinton (1998), as this interpretation best elucidates how the posterior regularization method we propose in Section 4 is a natural modification to standard EM. Neal and Hinton (1998) view EM as block coordinate ascent on a function that lower-bounds $\mathcal{L}(\theta)$. We form the lower bound, denoted $F(q, \theta)$, by applying Jensen's inequality to $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = \log \sum_{\mathbf{Y}} q(\mathbf{Y}) \frac{p_\theta(\mathbf{X}, \mathbf{Y})}{q(\mathbf{Y})} \geq \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{p_\theta(\mathbf{X}, \mathbf{Y})}{q(\mathbf{Y})} = F(q, \theta). \tag{3}$$

Splitting up the log terms, we can then rewrite $F(q, \theta)$ as:

$$\begin{aligned} F(q, \theta) &= \sum_{\mathbf{Y}} q(\mathbf{Y}) \log(p_\theta(\mathbf{X}) p_\theta(\mathbf{Y} \mid \mathbf{X})) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log q(\mathbf{Y}) \\ &= \mathcal{L}(\theta) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{q(\mathbf{Y})}{p_\theta(\mathbf{Y} \mid \mathbf{X})} \\ &= \mathcal{L}(\theta) - \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y} \mid \mathbf{X})). \end{aligned} \tag{4}$$

Based on this formula, we can view EM as performing coordinate ascent on $F(q, \theta)$. Starting from an initial parameter estimate θ^0, the algorithm iterates two block coordinate ascent steps until a convergence criterion is attained:

$$\mathbf{E}: q^{t+1} = \arg\max_{q} F(q, \theta^t) = \arg\min_{q} \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y} \mid \mathbf{X})) \tag{5}$$

$$\mathbf{M}: \theta^{t+1} = \arg\max_{\theta} F(q^{t+1}, \theta) = \arg\max_{\theta} \mathbf{E}_{q^{t+1}} [\log p_\theta(\mathbf{X}, \mathbf{Y})] \tag{6}$$

Note that the E-step just sets $q^{t+1}(\mathbf{Y}) = p_{\theta^t}(\mathbf{Y}|\mathbf{X})$, since it is an unconstrained minimization of a Kullback-Leibler divergence.

Figure 2 illustrates the large mismatch between an EM-trained DMV model and the empirical statistics of dependency types. We will eventually show that a parameter prior is insufficient to correct the mismatch, while posterior regularization is able to model dependency type statistics much more accurately.

3.2 Bayesian Learning

Recent advances in Bayesian inference methods have been applied to DMV grammar induction with varying levels of success. These approaches have focused on injecting additional linguistic intuition into the DMV by using a Dirichlet prior to sparsify parameters (Cohen et al., 2008; Headden III et al., 2009), or using logistic normal priors to tie parameters (Cohen et al., 2008; Cohen and Smith, 2009). In the following subsections, we'll discuss how these methods work and the types of improvements they are able to achieve. We'll present a more empirical comparison with these methods later, in Section 5.

3.2.1 Sparsity-Inducing Priors

One prior that has been extensively explored for DMV learning is the Dirichlet. More precisely, a product of Dirichlets: $p(\theta) = \prod_{A \in V_N} D(\theta_A; \alpha_A)$ where we consider the DMV as a PCFG, $G = (V_N, V_T, R, S)$ with V_N, V_T, and R a set of non-terminals, terminals, and rules, respectively, and S a start symbol. A description of the rules of the DMV as a PCFG can be found in Smith (2006). Each Dirichlet in this prior has the form:

$$D(\theta_A; \alpha_A) = \frac{1}{Z} \prod_{\beta : A \rightarrow \beta \in R} \theta_A(\beta)^{\alpha_{A \rightarrow \beta} - 1} \tag{7}$$

where Z is a normalization term and αs are hyperparameters.

The true posterior over the parameters, $p(\theta|\mathbf{X}) \propto \sum_{\mathbf{Y}} p(\mathbf{Y}, \mathbf{X}|\theta) p(\theta)$, is generally multi-modal and intractable to compute. The typical variational approximation is to define an approximate factored posterior over both parameters and latent variables, $q(\mathbf{Y}, \theta) = q(\mathbf{Y})q(\theta)$, and use mean field updates to minimize $\mathbf{KL}(q(\mathbf{Y})q(\theta)||p(\mathbf{Y}, \theta|\mathbf{X}))$. As shown by Kurihara and Sato (2004), with this type of prior, this can be accomplished efficiently. Assuming the hyperparameters of the prior are fixed, the coordinate descent algorithm for updating $q(\mathbf{Y}), q(\theta)$ is similar to EM. In the E-like-step, inference for $\mathbf{Y}$ is performed using the approximate mean parameters $\bar{\theta} = \mathbf{E}_q[\theta]$. The M-like-step, is a slight modification to the standard EM M-step, both shown below:

$$\textbf{EM } \mathbf{M}\text{-step}: \theta_A^{t+1}(\beta) \propto \ \mathbf{E}_{q^{t+1}}[\#_{A \rightarrow \beta}(\mathbf{Y})] \tag{8}$$

$$\textbf{Dirichlet } \mathbf{M}\text{-step}: \theta_A^{t+1}(\beta) \propto \ \exp(\psi(\mathbf{E}_{q^{t+1}}[\#_{A \rightarrow \beta}(\mathbf{Y})] + \alpha_{A \rightarrow \beta})) \tag{9}$$

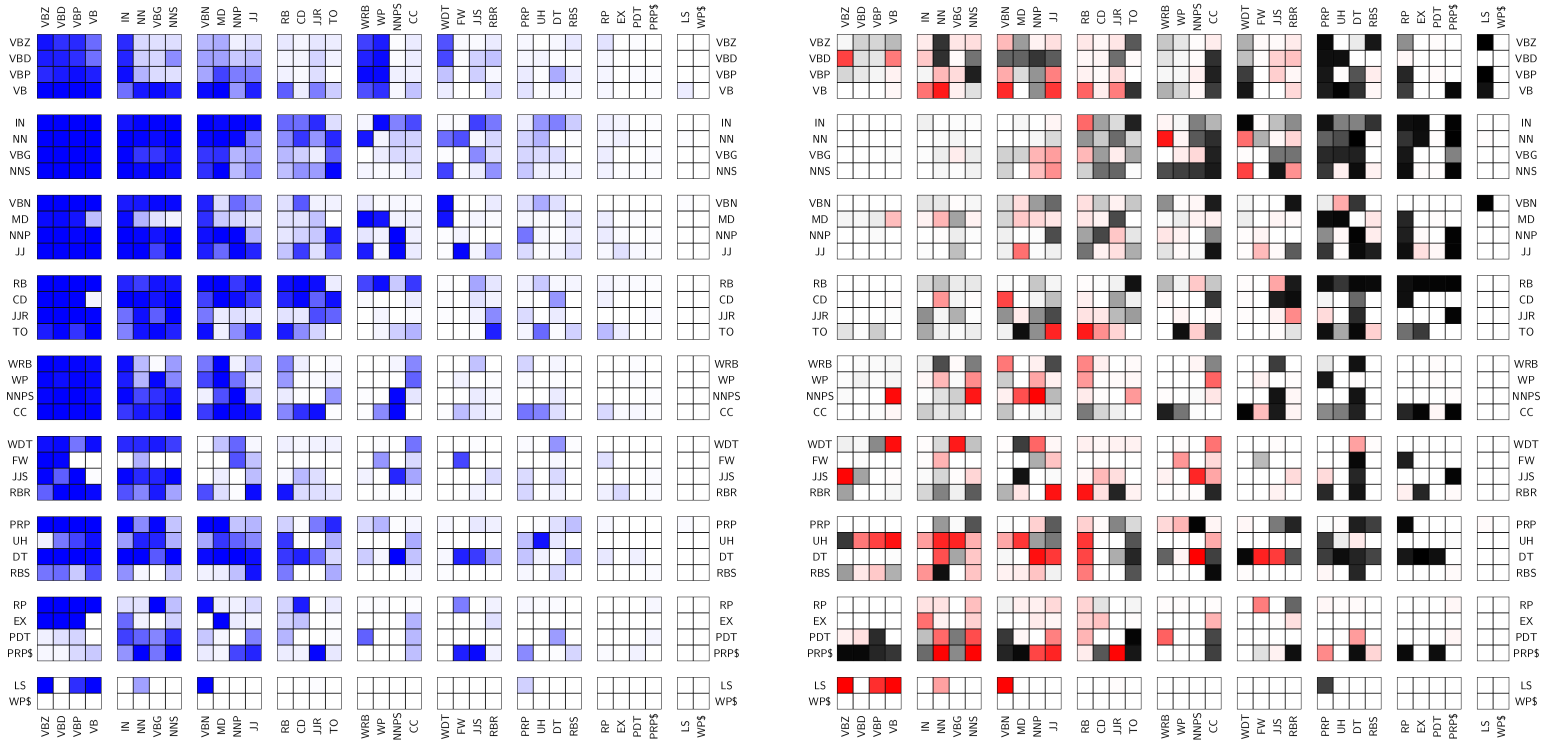


Figure 2: Comparison of parameters for a max likelihood DMV and an EM-trained DMV for English. Each square corresponds to a parent-child pair. Parent tags are listed across, child tags down. Parent tags are sorted left-to-right in descending order by the number of unique child tags they take. **Left**: Generated using max likelihood parameter settings (supervised). The saturation of a square with parent p and child c is determined by the maximum value of the posterior probability $p(p \mid c)$ observed in the entire English training corpus (Marcus et al., 1993). More saturated blue indicates higher probability. **Right**: Generated using EM parameter settings. Black indicates EM posteriors are too high, red too low. More saturation indicates more deviation. White indicates no deviation. There are significantly more black squares than red, especially towards the right, indicating that EM does not learn a sparse enough model.

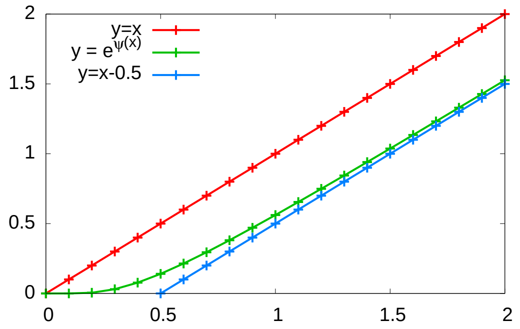


Figure 3: The digamma function.

where ψ is the digamma function. As Figure 3 illustrates, $\exp(\psi(x))$ is upper bounded by $y = x$. That is, it slightly discounts the value of x, though by no more than 0.5, as $y = x - 0.5$ lower bounds it. Thus, $\exp(\psi(x+\alpha))$ is similar to adding $\alpha - 0.5$ to x. For any $\alpha < 0.5$, this encourages parameter sparsity in the $\mathbf{Dirichlet\ M}$-step, since small θ will get squashed to zero by the digamma.

This Dirichlet prior method is applied in several previous works. Cohen et al. (2008) use this method for dependency parsing with the DMV and achieve improvements over basic EM. They set all hyperparameters to 0.25, resulting in a sparsifying prior (this is the method referred to as VB-Dirichlet in their work). Headden III et al. (2009) also use this method to train both the DMV and the E-DMV. However, they set all hyperparameters to 1, so their prior is not aimed at sparsifying. It nevertheless produces different results than standard EM because it sets parameters according to the mean of the posterior $q(\theta)$ instead of the mode.

In this paper we will refer to our own implementation of the VB-Dirichlet method of Cohen et al. (2008) as the"discounting Dirichlet" (DD) method. We will show experiments applying it to both the DMV and the E-DMV. In particular we will show that while it achieves parameter sparsity, this is not the optimal sparsity to aim for in dependency parsing. Intuitively, sparsity of $p_{child}(c \mid p, d)$ means requiring that each parent tag has few unique child tags. But note that, as the supervised grid in Figure 2 illustrates, some parents should be allowed many different types of children. For example, VBZ, VBD, VBP, VB, IN, NN, etc. all should be able to have non-zero $p_{child}(c \mid p, d)$ for many c. We will show that posterior regularization is one way to achieve a better type of sparsity.

3.2.2 Parameter-Tying Priors

In addition to Dirichlet, other types of priors have been applied, namely logistic normal priors (LN)(Cohen et al., 2008) and shared logistic normal priors (SLN) (Cohen and Smith, 2009). While the DD aims to induce parameter sparsity, LN and SLN aim to tie parameters together. Essentially, this has a similar goal to sparsity-inducing methods in that it posits a more concise explanation for the grammar of a language. That is, it suggests that POS tags share certain properties and so the grammar is not really as ambiguous as the full range of possible parameter settings would suggest.

The LN prior has the form $p(\theta) = \prod_{A \in V_N} \mathcal{N}(\mu_A, \Sigma_A)$ where μ_A is a mean vector and Σ_A is a covariance matrix for a normal distribution over the PCFG rules with lefthand side A. The Σ_A allow rules with identical lefthand sides to co-vary, effectively tying these parameters. For example, LN can tie the parameters $p_{child}(c_1 \mid p, d)$ and $p_{child}(c_2 \mid p, d)$. The SLN prior extends the capabilities of the LN prior by allowing any arbitrary parameters to be tied. In this case, parameters such as $p_{child}(c \mid p_1, d)$ and $p_{child}(c \mid p_2, d)$ can be tied even though they correspond to PCGF rules with different lefthand sides. We compare in the experimental section against some results from using LN and SLN and show that our posterior regularization method produces higher accuracy results.

As a side note, we mention that Headden III et al. (2009) also implements a sort of parameter tying for the E-DMV through a backoff distribution on child probabilities. The form of the backoff was introduced in Equation 2. The way Headden III et al. (2009) choose the weighting $(1 - \lambda)$ for the backoff is through a Dirichlet prior. To capture the intuition that events seen fewer times should be more strongly smoothed, this prior has hyperparameter value K for the standard child probability and value $2K$ for the backoff probability, where K is the number of PCFG rules with a particular nonterminal on the left-hand side. This ensures that the backoff probability is only ignored when enough examples of the full child probability have been seen. The prior favors the backoff 2 to 1, which is why in our approximation of this scheme we use weight $\lambda = 1/3$.

3.3 Other Learning Approaches

Several additional training alternatives have been proposed besides Bayesian methods. In particular, we will briefly describe here three such methods: contrastive estimation (CE), skewed deterministic annealing (SDA), and structural annealing (SA).

The first approach, contrastive estimation (CE), has been applied to several applications for training log linear models on unlabeled data (Smith and Eisner, 2005b,a). The basic idea is to maximize the following:

$$\log \prod_i \frac{\sum_{\mathbf{y} \in \mathbf{Y}} \exp(\theta \cdot f(\mathbf{x}^{(i)}, \mathbf{y}))}{\sum_{(\mathbf{x},\mathbf{y}) \in N(\mathbf{x}^{(i)}) \times Y} \exp(\theta \cdot f(\mathbf{x}, \mathbf{y}))} \tag{10}$$

where f is some vector of feature functions, and $N(\mathbf{x}^{(i)})$ is a set of $\mathbf{x}$ that are in the "neighborhood" of $\mathbf{x}^{(i)}$. The intuition behind this method is that if a person chose to produce $\mathbf{x}^{(i)}$ out of all the possible $\mathbf{x}$ in $N(\mathbf{x}^{(i)})$, then we want to learn a model that assigns higher value to $\mathbf{x}^{(i)}$ (the numerator in Equation 10) than to these other $\mathbf{x}$. Restricting to a neighborhood is necessary for tractability, and the choice of neighborhood can encode linguistic knowledge. For example, for dependency parsing Smith and Eisner (2005a) formed neighborhoods by deleting any one word from $\mathbf{x}^{(i)}$, or transposing any two words.

Two other non-Bayesian approaches of note are skewed deterministic annealing (SDA) and structural annealing (SA) (Smith and Eisner, 2006). SDA biases towards shorter dependency links as in the K&M initializer, and flattens the likelihood function to alleviate the difficulty of escaping local maxima. Alternatively, SA biases strongly toward short dependency links in early iterations, then relaxes this constraint over time.

We present an empirical comparison to the three methods from this section in Section 5 and show we can often achieve superior performance with posterior regularization.

4. Learning with Sparse Posteriors

At a high level, we would like to penalize models $p_\theta(\mathbf{Y}|\mathbf{X})$ that predict a large number of distinct dependency types. For hard assignments, this quantity is easy and intuitive to measure. Define an edge type as the pair composed of parent POS and child POS. Given a corpus with parse trees, we are interested in the number of distinct types of edges used in the labeling. Section 4.2 describes how to extend this definition to distributions over parse trees, by viewing it as a mixed-norm. Having a small number of distinct dependency types is a kind of sparsity structure we would like to impose on the model.

To achieve this desired sparsity, we use the posterior regularization (PR) framework (Graça et al., 2007). PR is closely related to generalized expectation constraints (Mann and McCallum, 2007, 2008; Bellare et al., 2009), and is also indirectly related to a Bayesian view of learning with constraints on posteriors (Liang et al., 2009). The PR framework uses constraints on posterior expectations to help guide parameter estimation. It allows for tractable learning and inference even when the constraints it enforces would be intractable to encode directly as additional model parameters or structure. In particular, PR allows a natural representation of the dependency sparsity constraints based on the type count ϕ_{cpi} described above.

4.1 Posterior Regularization

PR can be seen as a penalty on the standard marginal log-likelihood objective, which we define first as:

$$\textbf{Likelihood objective:}\ \mathcal{L}(\theta) = \log p_\theta(\mathbf{X}) + \log p(\theta) \\ = \sum_{\mathbf{x}\in\mathbf{X}} [\log \sum_{\mathbf{y}} p_\theta(\mathbf{x},\mathbf{y})] + \log p(\theta) \tag{11}$$

where θ represents the model parameters, $p(\theta)$ is a (optional) prior probability on the parameters, and the sum is over the unlabeled sample data. Recall that we use $\mathbf{x}$ to denote a single sentence's POS tags, and $\mathbf{y}$ to denote a single hidden parse tree.

Here we present the penalty version of PR; Ganchev et al. (to appear 2010) describe a constraint-set version of PR and give more details. In PR, the desired bias is specified with a penalty on expectations of features ϕ. For any distribution q over latent variables, we can define a penalty as the β-norm of the feature expectations:

$$||\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]||_\beta \tag{12}$$

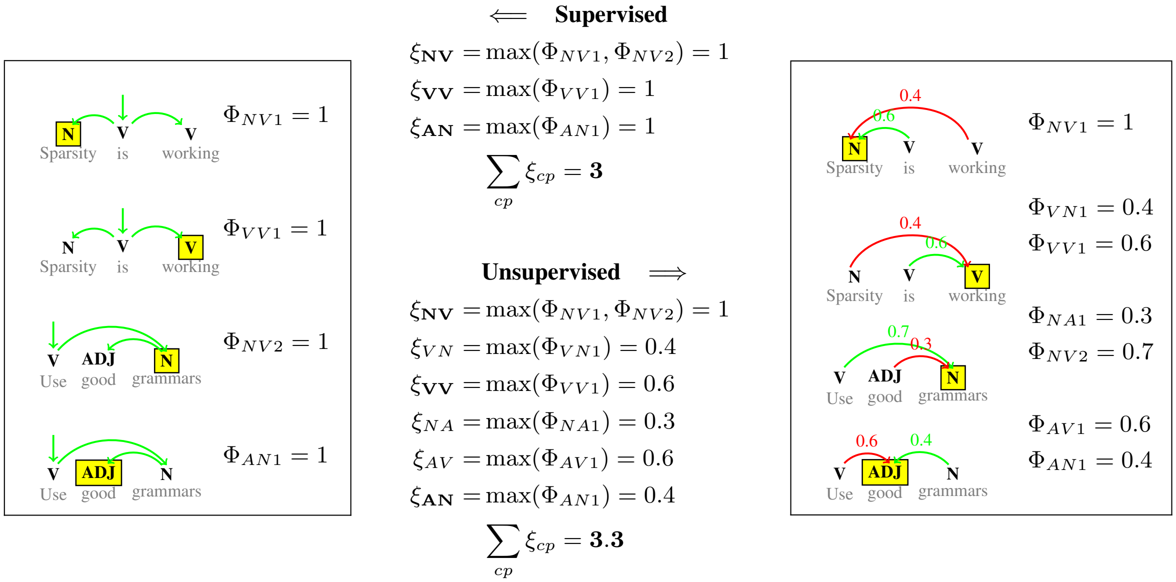
where $\mathbf{Y}$ represents an assignment of parse trees for all sentences in the corpus $\mathbf{X}$. For computational tractability, rather than penalizing the model's posteriors directly, we use an auxiliary distribution, and penalize the marginal log-likelihood of a model by the KL-divergence and penalty term with respect to q. For a fixed set of model parameters θ the PR penalty term we will use is given by:

$$\textbf{Penalty term:}\ \min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma\, ||\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]||_\beta \tag{13}$$

As we will see, using an auxiliary distribution q will make the final objective easier to optimize. Ganchev et al. (to appear 2010) describe how to compute this penalty term in general, but we will defer that explanation to Section 4.2 when we describe our particular penalty term. The PR framework seeks to maximize:

$$\textbf{PR objective:}\ \mathcal{L}(\theta) - \min_q \Big[\mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma\, ||\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]||_\beta\Big]. \tag{14}$$

The objective in Equation 14 can be optimized by a variant of the EM (Dempster et al., 1977) algorithm used to optimize the objective in Equation 11.



$\Longleftarrow$ **Supervised**

$$\begin{aligned}
\xi_{\mathbf{NV}} &= \max(\Phi_{NV1}, \Phi_{NV2}) = 1\\
\xi_{\mathbf{VV}} &= \max(\Phi_{VV1}) = 1\\
\xi_{\mathbf{AN}} &= \max(\Phi_{AN1}) = 1\\
&\sum_{cp} \xi_{cp} = \mathbf{3}
\end{aligned}$$

Unsupervised $\Longrightarrow$

$$\begin{aligned}
\xi_{\mathbf{NV}} &= \max(\Phi_{NV1}, \Phi_{NV2}) = 1\\
\xi_{VN} &= \max(\Phi_{VN1}) = 0.4\\
\xi_{\mathbf{VV}} &= \max(\Phi_{VV1}) = 0.6\\
\xi_{NA} &= \max(\Phi_{NA1}) = 0.3\\
\xi_{AV} &= \max(\Phi_{AV1}) = 0.6\\
\xi_{\mathbf{AN}} &= \max(\Phi_{AN1}) = 0.4\\
&\sum_{cp} \xi_{cp} = \mathbf{3.3}
\end{aligned}$$

Figure 4: The ℓ_1/ℓ_∞ regularization term for a toy example. Let $\Phi_{cpi} = \mathbf{E}_q[\phi_{cpi}]$. For simplicity we ignore the root $\rightarrow c$ edges here, though in our experiments we incorporate their probabilities also. **Left**: Two gold parse trees with two (non-root) children each. Edges in the trees have probability 1, and all other edges probability 0, resulting in an ℓ_1/ℓ_∞ of 3. **Right**: In the unsupervised setting, instead of gold trees we have a posterior distribution over parents for each child. Given the distribution shown, the ℓ_1/ℓ_∞ is 3.3. Since real grammars tend to have few edge types, it makes sense that the ℓ_1/ℓ_∞ of a set of supervised trees will be small. Thus, using regularization to force ℓ_1/ℓ_∞ to be small for posterior distributions should push these distributions closer to the gold.

4.2 ℓ_1/ℓ_∞ Regularization

We now define precisely how to count dependency types, which will allow us to specify different kinds of dependeny sparsity. For each child tag c, let i range over some arbitrary enumeration of all occurrences of c in the corpus, and let p be another tag. The indicator $\phi_{cpi}(\mathbf{X}, \mathbf{Y})$ has value 1 if p is the tag of the parent of the ith occurrence of c, and value 0 otherwise. The number of unique dependency types is then given by:

$$\sum_{cp} \max_i \phi_{cpi}(\mathbf{X}, \mathbf{Y}), \tag{15}$$

where we sum over child-parent types cp, computing the maximum (logical or) over possible occurrences of $c \leftarrow p$ dependencies. Note that there is an asymmetry in this way of counting types: occurrences of the child type c are enumerated with i, but all occurrences of the parent type p are or-ed in ϕ_{cpi}, that is, ϕ_{cpi} is 1 if *any* occurrence of tag p is the parent of the ith occurrence of tag c, we will refer PR training with this constraint as PR-AS.

Instead of counting pairs of a child token and a parent type, we could instead have counted pairs of a child token and a parent token by letting p range over all *tokens* rather than *types*. In that case, each potential dependency would correspond to a different indicator ϕ_{cpij}, and the penalty would be symmetric with respect to parents and children, , we will refer PR training with this constraint as PR-S. Because of the way the model is parameterized, an asymmetry in the other direction (child-type, parent-token) does not make sense: the sum over all child types is always 1, so the penalty term would be a constant.

Both approaches perform very well, however one approach is not clearly better than the other when compared across the twelve languages. So, we report results for both versions on the results section.

Equation 15 can be viewed as a mixed-norm penalty on the features ϕ_{cpi}. More precisely, we will penalize the following quantity: the sum (ℓ_1 norm) over c of the maximum (ℓ_∞ norm) over occurrences of c of the posterior probability of selecting a parent with tag p for that child. This falls in the PR framework described in Section 4.1. Figure 4 shows a toy example of how to compute the ℓ_1/ℓ_∞ regularization term. In order to compute the value of the PR objective and also to optimize it, we need to compute the projection

$$\arg\min_{q} \mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma \sum_{cp} \max_{i} \mathbf{E}_q[\phi_{cpi}(\mathbf{X},\mathbf{Y})]. \tag{16}$$

Which can equivalently be written as:

$$\begin{aligned} \min_{q(\mathbf{Y}),\xi_{cp}} \quad & \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma \sum_{cp} \xi_{cp} \\ \text{s. t.} \quad & \xi_{cp} \leq \mathbf{E}_q[\phi_{cpi}(\mathbf{X},\mathbf{Y})] \ \ \forall c,p,i \end{aligned} \tag{17}$$

where σ is the strength of the regularization, and ξ_{cp} corresponds to the maximum expectation of ϕ_{cpi} over all c and p. Note that the projection problem is convex in q and can be solved efficiently in the dual (just as for the maximum entropy/log linear model fitting). The formulation of Equation 17 makes the derivation of the dual easier (see Ganchev et al. (to appear 2010) for a derivation of the dual in the general case). The dual of the projection problem is a fairly simple convex problem:

$$\begin{aligned} \min_{\lambda \geq 0} \quad & \log \left(\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{X},\mathbf{Y})) \right) \\ & \text{s. t.} \quad \sum_{i} \lambda_{cpi} \leq \sigma \end{aligned} \tag{18}$$

where ϕ is the vector of feature values ϕ_{cpi} for assignment $\mathbf{Y}$ of parse trees to the entire corpus $\mathbf{X}$, and $\boldsymbol{\lambda}$ is the vector of dual parameters λ_{cpi}. The primal parameters are related to the dual by the equation $q(\mathbf{Y}) \propto p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{X},\mathbf{Y}))$, and can in this form be computed via projected gradient, as described by Bertsekas (1995). Note that projection onto the simplex constraints can be done very efficiently as described in Bertsekas (1995).

When σ is zero, the projection is an identity mapping and the algorithm reduces to EM. As $\sigma \to \infty$, the constraints force the posterior probability of parent tag given child tag to be uniform. For intermediate values of σ, the constraints work to decrease the confidence of the highest probability parent tags for each child instance. For parent tags that are supported by many high-probability

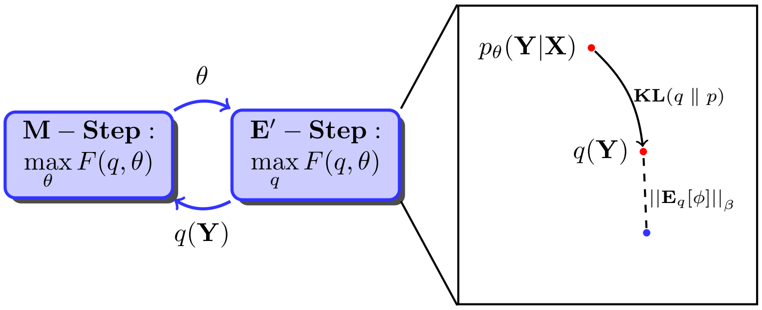

θ
$\mathbf{M}-\mathbf{Step}:$ $\max_{\theta} F(q,\theta)$
$\mathbf{E'}-\mathbf{Step}:$ $\max_{q} F(q,\theta)$
$q(\mathbf{Y})$
$p_\theta(\mathbf{Y}|\mathbf{X})$
$\mathbf{KL}(q \parallel p)$
$q(\mathbf{Y})$
$||\mathbf{E}_q[\phi]||_\beta$

Figure 5: Modified EM for optimizing PR objective.

instances, this pressure is distributed among many instances and has little effect. For parent tags that are supported by few high-probability instances however, the probability of these instances is more severely reduced, which can (after several iterations of the algorithm) effectively eliminate that parent tag as a possibility for the given child tag.

4.3 Optimization Algorithms

The optimization algorithm for the PR objective uses a minorization-maximization procedure akin to EM.

The PR objective (Equation 14) is

$$J(\theta) = \max_q F'(q,\theta) = \mathcal{L}(\theta) - \min_q \left[\mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \ + \ \sigma\, ||\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]||_\beta\right] \qquad (19)$$

This objective can be optimized by modifying the E-step of EM to include the β-norm penalty:

$$\mathbf{E'}: q^{t+1} = \arg\max_q F'(q,\theta^t) = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})) \ + \ \sigma\, ||\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]||_\beta \qquad (20)$$

The projected posteriors $q^{t+1}(\mathbf{Y})$ are then used to compute sufficient statistics and update the model's parameters in the M-step, which remains unchanged, as in Equation 6. This scheme is illustrated in Figure 5. The following proposition is adapted from Ganchev et al. (to appear 2010) who have a version for hard constraints.

Proposition 4.1 *For the modified EM algorithm illustrated in Figure 5, which iterates the modified E-step (Equation 20) with the normal M-step (Equation 6), monotonically increases the PR objective:* $J(\theta^{t+1}) \geq J(\theta^t)$.

Proof The proof is analogous to the proof of monotonic increase of the standard EM objective. Essentially,

$$J(\theta^{t+1}) = F'(q^{t+2},\theta^{t+1}) \geq F'(q^{t+1},\theta^{t+1}) \geq F'(q^{t+1},\theta^t) = J(\theta^t).$$

The two inequalities are ensured by the $\mathbf{E'}$-step and M-step. $\mathbf{E'}$-step sets $q^{t+1} = \arg\max_q F'(q,\theta^t)$, hence $J(\theta^t) = F'(q^{t+1},\theta^t)$. The M-step sets $\theta^{t+1} = \arg\max_\theta F'(q^{t+1},\theta)$, hence $F'(q^{t+1},\theta^{t+1}) \geq$

	Bg	Cz	De	Dk	En	Es	Jp	Nl	Pt	Se	Si	Tr
tags	11	58	51	24	34	17	74	162	19	31	26	28
sentences	4711	24494	13092	1770	5358	428	12286	6578	2415	3467	476	3331
word types	11342	40156	19606	5750	10238	2610	1918	10695	7279	7633	2799	10480
word tokens	27313	139360	76843	10899	37066	2444	43209	42743	14438	23250	3025	17682

Table 1: Corpus statistics for sentences with lengths ≤ 10, after stripping punctuation. Bg stands for Bulgarian, Cz for Czech, De for German, Dk for Danish, En for English, Es for Spanish, Jp for Japanese, Nl for Dutch, Pt for Portuguese, Se for Swedish, Sl for Slovene, and Tr for Turkish.

$F'(q^{t+1}, \theta^t)$. Finally, $J(\theta^{t+1}) = \max_q F'(q, \theta^{t+1}) \geq F(q^{t+1}, \theta^{t+1})$ ■

As for standard EM, to prove that coordinate ascent on $F'(q, \theta)$ converges to stationary points of $J(\theta)$, we need to make additional assumptions on the regularity of the likelihood function and boundedness of the parameter space as in Tseng (2004). This analysis can be easily extended to our setting, but is beyond the scope of the current paper.

5. Experiments

5.1 Corpora

We evaluate our models on 12 languages—the English Penn Treebank (Marcus et al., 1993) and 11 languages from the CoNLL X shared task: Bulgarian [Bg] (Simov et al., 2002), Czech [Cz] (Bohomovà et al., 2001), German [De] (Brants et al., 2002), Danish [Dk] (Kromann et al., 2003), Spanish [Es] (Civit and Martí, 2004), Japanese [Jp] (Kawata and Bartels, 2000), Dutch [Nl] (Van der Beek et al., 2002), Portuguese [Pt] (Afonso et al., 2002), Swedish [Se] (Nilsson and Hall, 2005), Slovene [Sl] (Džeroski et al., 2006), and Turkish [Tr] (Oflazer et al., 2003). For English we train on sections 2-21 of the Penn Treebank and test on section 23. For the other languages our train and test sets are exactly those from the CoNLL X shared task. Following the example of Smith and Eisner (2006), we strip punctuation from the sentences and keep only those sentences that are of length ≤ 10. Table 1 shows the size of the different training corpora after this filtering.

5.2 Results on English

We start with a comparison between EM and the two sparsity-inducing methods, PR and the discounting Dirichlet prior (DD), on the English corpus. For all models we use the "harmonic" K&M initializer and then train for 100 iterations. At the end of training, each model is evaluated on the test set using the Viterbi (best) parse. Before evaluating, we smooth the resulting models by adding e^{-10} to each learned parameter, merely to remove the chance of zero probabilities for unseen events. (We did not bother to tune this value at all as it makes very little difference for final parses.) We score models by their attachment accuracy — the fraction of words assigned the correct parent. We compare the performance of all training procedures both on the original DMV model as well as on the extended model E-DMV. In the case of E-DMV, we set the smoothing for child probabilities

		DD $\alpha =$			
	EM	1	0.25	0.1	0.01
DMV	45.8	42.2	**46.4**	45.2	45.4
2-1	45.1	42.0	46.0	45.9	44.9
2-2	**54.4**	42.0	43.3	52.5	51.5
3-3	55.3	42.8	47.1	53.5	52.1
4-4	55.1	42.9	47.1	53.6	51.7

Table 2: Directed attachment accuracy results on the test corpus (for sentences of lengths $\leq$ 10, no punctuation). The second column gives EM results, and the other columns are DD results for different settings of the hyperparameter α. The second row is for the basic DMV model, and the other rows are E-DMV models represented by their valencies (V_c-V_s). Bold represents the best parameter setting both for the DMV model and the E-DMV model.

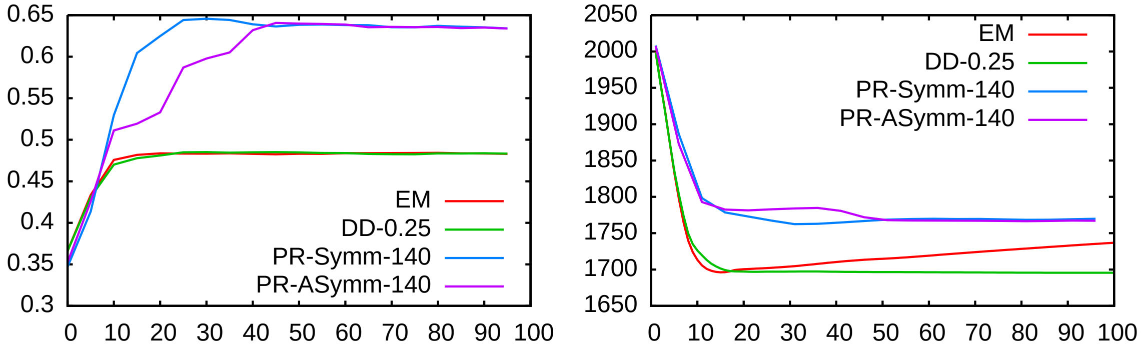
to 0.66, based on the hyperparameter used in Headden III et al. (2009). We keep smoothing fixed across languages and model configurations to reduce the number of parameters that need to be chosen. Following Cohen et al. (2008) we search for the best discounting parameter α for DD training. We tried we tried 5 different values for α: $\{0.01, 0.1, 0.25, 1\}$.

Table 2 shows the directed accuracy for both the DMV and the E-DMV models trained using EM and DD. We see in Table 2 that the extended model generally outperforms the DMV, for both EM and DD. However, we also see that DD does not always help: for all valences tried for the E-DMV except $(V_C, V_S) = (2, 1)$, the EM models perform better. This contrasts with the findings of Headden III et al. (2009), potentially due to the simplified smoothing that we implemented, and a difference in the stopping criterion — we ran our model for 100 iterations, while Headden III et al. (2009) ran until likelihood on a heldout development set converged. Another explanation is that there are interactions of the model initialization and training. Headden III et al. (2009) use the RandomP initialization described in Subsection 2.3 while we use the harmonic K&M initializer. Comparing the performance of the training methods, we see that for the DMV model, DD training performs better and the best hyperparameter setting is 0.25 which is the same best parameter found by Cohen et al. (2008). The performance of our implementation of the DD is slightly lower than the one reported in that paper, probably due to different stopping criteria during training.

A comparison between EM and PR for both DMV and E-DMV are shown in Table 3. We searched over six different regularization strengths $\{80, 100, 120, 140, 160, 180\}$, for both the PR-S (symmetric constraint) and PR-AS (asymmetric constraint) formulations. As with Table 2, the results in Table 3 show attachment accuracy for Viterbi decoding. In Graça et al. (to appear 2010), the authors found that for PR, projecting at decoding consistently improved results on the task of word alignment. Consequently, they always compute the projected distribution q and decode using q rather than the model distribution. In this work, we found that projecting at decode time produced worse results. The regularization strength parameter σ is corpus-dependent and in particular depends on the size of the corpus. Because the test corpus is much smaller than the training corpus,

Model	EM	PR-S						PR-AS					
	DMV												
σ		80	100	120	140	160	180	80	100	120	140	160	180
	45.8	60.1	60.8	61.1	**62.1**	60.8	60.2	40.4	53.8	61.7	61.9	54.7	54.3
V_c-V_s	E-DMV												
2-1	45.1	61.1	**62.7**	62.5	62.0	61.8	60.2	40.5	54.5	61.7	62.1	54.4	62.0
2-2	54.4	**62.9**	57.3	57.4	57.4	56.7	59.2	56.2	56.3	56.8	57.0	58.5	58.7
3-3	55.3	59.4	60.4	61.1	**64.3**	63.4	62.6	60.0	60.0	61.4	63.9	64.0	59.3
4-4	55.1	61.0	62.8	64.1	63.5	64.1	59.4	59.9	60.5	**64.4**	64.1	58.1	59.7

Table 3: Directed attachment accuracy results on the test corpus. Bold represents the best parameter setting for the DMV model and for each of the E-DMV models. The first column contains the V_c-$V_s$used. Columns represent different σ for both constraints PR-S on the left and PR-AS on the right.



(a) Accuracy

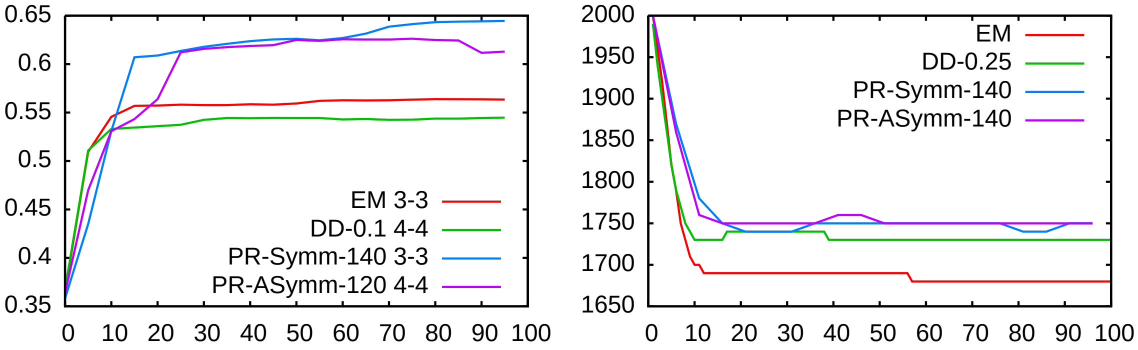
(b) Negative log-likelihood

Figure 6: Accuracy and negative log likelihood on heldout development data as a function of the training iteration for the DMV.

using the same σ for training and testing might be the wrong option. Thus, in this paper we do not project at decode time.

A first observation based on Table 3 is that PR-S generally performs better than the PR-AS. Furthermore, PR-S seem less sensitive to the particular regularization strength. Comparing PR-S to EM, PR-Sis always better, independent of the particular σ, with improvements ranging from 4% to 16%. The PR-AS constraints are also always better than EM for each model configuration and for all but two different parameter configurations. Note that the optimal parameter σ depends on the particular model configuration (V_c-V_s).

Figure 6 shows how accuracy and negative log-likelihood change on a heldout development corpus for the DMV. We see that both with EM and DD the models tend to converge after only 10 iterations, while for the PR training it takes 20 to 30 iterations. PR also seems to overfit and experi-



(a) (b)

Figure 7: Directed accuracy and negative log likelihood on held-out development data as a function of the training iteration for the E-DMV model with the best parameter setting.

ence degrading performance on the test set after 40 iterations. We see in Figure 6 that accuracy and likelihood tend to correlate well and could potentially be used as a stopping criterion for training. In fact, most prior work uses this criterion to stop training. This appears to work for the DMV model. Figure 7 shows similar graphs for the E-DMV model. We see Figure 7 that the likelihood-accuracy correlation does not hold as cleanly. In fact the PR-AS model seem to be improving accuracy after 100 iterations, while the optimal likelihood is achieved around iteration 20. Stopping at iteration 20 rather than 100 would have reduced accuracy by about 4%. For the rest of this paper we run all our experiences for 100 iterations and report results for the model obtained at the end of training.

Also we note that we found no correlation between the development likelihood and the best setting for the constraint strength when training with PR, which makes it harder to pick these parameters in an unsupervised setting. We also cannot use likelihood to choose between different valencies for the E-DMV model, since the likelihoods are not comparable.

5.3 Comparison with Previous Work

In this section we compare the performance of different models described in the literature for unsupervised dependency parsing. Table 4 presents the accuracy values reported in various previous papers and the values for approaches tried in this paper. We would like to stress that the setup is not identical for all experiments. For instance, normally the stopping criteria for training is different. While we train all our models for 100 iterations, most other works use some kind of convergence criteria to stop training. Moreover, there are likely differences regarding other implementation details.

We start by comparing the effects of different initialization procedures. Although orthogonal to the learning procedure used, these differences are significant when comparing to previous work. We compare the results for the DMV with the different initializations described in Section 2.3. All the different approaches significantly beat the K&M initialization by about 10% in accuracy. There are some differences in the setup of the different approaches: the model initialized with RandomP described in Headden III et al. (2009) is trained using DD with a hyperparameter of 1, while all the other models are trained using EM. Additionally, the models from Spitkovsky et al. (2010) use a larger amount of data.

The next comparison we make is between the smoothing approach described in Headden III et al. (2009) and the simpler implementation done in this work. Again, although the training methods and the initialization differs we see that the smoothing performed by Headden III et al. (2009) probably increases the accuracy of that model by around 5.5% over our implementation of smoothing (see entries 1, 2 7, and 8).

Entries 9 to 20 compare different training approaches for the basic DMV. Entry 9 corresponds to training the model with DD with the best hyperparameter setting. Entries 10 and 11 correspond to training with PR under the two types of sparsity constraints. Entries 12 and 13 use the logistic normal prior (Cohen et al., 2008) and we report the results from the paper using Viterbi decoding. Entries 14, 15, 16, and 17 correspond to the different shared logistic normal priors (Cohen and Smith, 2009). These values are for MBR decoding since the authors don't report values for Viterbi decoding. This gives some advantage to these entries, since according to the authors MBR decoding always outperforms Viterbi decoding. Finally, entries 18, 19, and 20 represent the best value for the three learning approaches contrastive estimation (CE), skewed deterministic annealing (SDA), and structural annealing (SA) proposed by Smith (2006). For these entries we report the best values found using supervised model selection. Out of all of these methods, the models trained using PR with the sparsity inducing constraints achieve the best results, the symmetric prior being the best. The results are similar to the best shared logistic normal prior when tested on sentences of length up to ten, but when tested on longer sentences the PR trained models perform significantly better then all other approaches.

The last block of results, entries 21 to 28, shows how a variety of learning methods compare on E-DMVs. Entries 21 to 24 compare our implementation of the three different learning approaches, EM, DD, and PR with both types of constraints. Model selection in these cases is supervised, based on accuracy for the $\leq$ 10 test data. PR significantly outperforms the other two approaches. In particular the PR-S constraints perform the best with an average of 10% improvement over EM and DD on sentences of lengths $\leq$ 10, and an even bigger improvement for longer sentences. In entries 25 to 28 we also compare with the original extended model of McClosky (2008) and with the smoothed extended model proposed by Headden III et al. (2009). The best model is the E-DMV with smoothing on the child probability as described by Headden III et al. (2009). It beats the E-DMV trained with PR-S by a small amount. This small difference, 0.7%, is much smaller than the gains from using the random initialization and the better smoothing distribution. Thus, we believe that training the same model with random initialization, better child probability smoothing, and the PR constraints would in fact produce the best results. We leave this as future work.

Finally we would like to note that Table **??** doesn't report results for the papers that use extra information. Bamely, Headden III et al. (2009) reports the best result published so far, 68.8, for the test set with sentences of lengths ≤ 10, when using lexical information. Also, Cohen and Smith (2009) reports accuracies of 62.0, 48.0, and 42.2 for sentences of lengths ≤ 10, sentences of lengths ≤ 20, and all sentences, respectively, when using multilingual information. This result for sentences of length ≤ 10 is equal to our best result, but is inferior to our results on longer sentences.

	Init	Model	Directed			Undirected		
			≤ 10	≤ 20	all	≤ 10	≤ 20	all
Model Initialization								
1	K&M	DMV	45.8	40.2	35.9	63.4	58.0	54.2
2	RandomP	DMV	55.7(8.0)					
3	BS	Ad-Hoc @15	55.5	44.3	39.2			
4	BS	Ad-Hoc @45	55.1	44.4	39.4			
5	LsM	Ad-Hoc @15	56.2	48.2	44.1			
6	LP	Hybrid @45	**57.1**	**48.7**	**45.0**			
Smoothing effects								
7	RandomP	E-DMV(2,1) smoothed distribution	**61.2(1.2)**					
8	K&M	E-DMV(2,1)	45.1	38.7	34.0	62.7	56.9	52.7
DMV								
9	K&M	DD (0.25)	46.4	40.9	36.5	64.0	58.6	54.8
10	K&M	PR-Symm 140	**62.0**	**53.8**	**49.1**	70.0	**62.6**	**58.4**
11	K&M	PR-ASymm 140	61.9	53.3	48.6	**70.2**	62.3	58.1
12	K&M	LN I	56.6	43.3	37.4			
13	K&M	LN families	59.3	45.1	39.0			
14	K&M	SLN TieV	60.2	46.2	40.0			
15	K&M	SLN TieN	60.2	46.7	40.9			
16	K&M	SLN TieV & N	61.3	47.4	41.4			
17	K&M	SLN TieA	59.9	45.8	40.9			
18	K&M	CE	48.7			64.9		
19	K&M	SDA	46.7			64.3		
20	K&M	SA	51.5			67.9		
E-DMV								
21	K&M	EM 3-3	55.3	46.4	42.6	69.0	61.9	58.3
22	K&M	DD 4-4 (0.1)	53.6	43.8	39.6	67.5	59.0	54.9
23	K&M	PR-Symm 3-3 140	64.3	**57.2**	**53.3**	**69.7**	**60.7**	**56.0**
24	K&M	PR-ASymm 4-4 120	60.5	49.8	44.6	67.9	59.3	54.8
25	K&M	EM 2-2	56.5			**69.7**		
26	RandomP	DD 2-2 (1)	53.3(7.1)					
27	RandomP	DD 2-2 (1) smoothed-skip-val	62.1(1.9)					
28	RandomP	DD 1-1 (1) smoothed-skip-head	**65.0(5.7)**					

Table 4: Comparison with previous published results. Results for entries 3, 4, 5, and 6 are taken from Spitkovsky et al. (2010), entries 2, 7, 26, 27, and 28 are taken from Headden III et al. (2009), entry 25 is taken from McClosky (2008), entries 12 and 13 are taken from Cohen et al. (2008), entries 14, 15, 16, and 17 are taken from Cohen and Smith (2009) and entries 18, 19, and 20 are taken from Smith (2006). See section text for details of the comparison.

5.4 Multilingual Results

A grammar induction algorithm is more interesting if it works on a variety of languages. Otherwise, the algorithm might just encode a lot of language-specific information. In this section, we compare several models and learning methods on twelve different languages to test their generalization capabilities. We do not want to assume that a user would have parsed corpora in each language, so we do not include a supervised search over model parameters for all languages as part of the evaluation process. Consequently, we use the following setup: for each model, basic DMV and the four E-DMV complexities we experimented with in the previous sections, pick the best configuration found for English according to its accuracy on the ≤ 10 test set, and use it across the other eleven languages. This might not select the ideal parameters for any particular language, but provides a more realistic test setting: a user has available a labeled corpus in one language, and would like to induce grammars for other languages of interest.

For the PR approach, since the ideal strength is related to corpus size, we try two different approaches. The first is to use exactly the same strength with other languages as used for English. The second approach is to scale the strength by the number of tokens in each corpus. In this case, the strength, σ_x, for a particular language was found by the following formula: $\sigma_x = \sigma_{en} * |tokens_{en}|/|tokens_x|$, where σ_{en} is the best strength for English, $|tokens_{en}|$ is the number of tokens of the English corpus, and $|tokens_x|$ is the number of tokens in language x. This scaling is an approximation that attempts to require a similar amount of sparsity for each language.

Table 5 shows the performance for all models and training procedures for the 12 different languages. Figure 8 illustrates the differences between the EM training and the different sparsity inducing training methods for the DMV. The zero line in Figure 8 corresponds to performance equal to EM. We see that the sparsifying methods tend to improve over EM most of the time. The average improvements are shown in the key of Figure 8. Figure 9 shows a similar comparison of the PR methods with respect to a DD learning baseline. We see in Figure 9 that PR is better than DD for most languages.

Figure 10 compares the different sparsity approaches. On the left we compare PR-S versus PR-AS without scaling. PR-AS beats PR-S in 9 out of 12 cases, though the average increase is only 1.5%. On the right we compare PR-S without scaling versus PR-S with scaling. The average improvement of the unscaled version is bigger for both constraints.

Figure 11 compares the differences of each training method against EM training using the E-DMV model with the best setting found for English. The results are similar to those for the DMV model with the biggest difference being that DD training performs worst. Both PR-S and PR-AS perform better than EM in most cases and the average improvement is even bigger than for the DMV model.

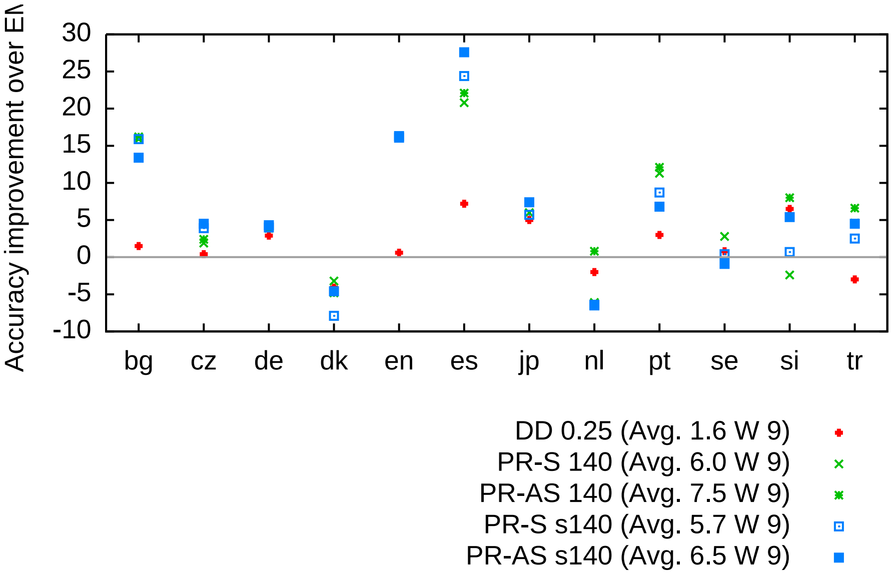


Figure 8: Difference in accuracy between the sparsity inducing training methods and EM training for the DMV model across the 12 languages. Avg. - Average improvement over EM. W - Number of languages better than EM.

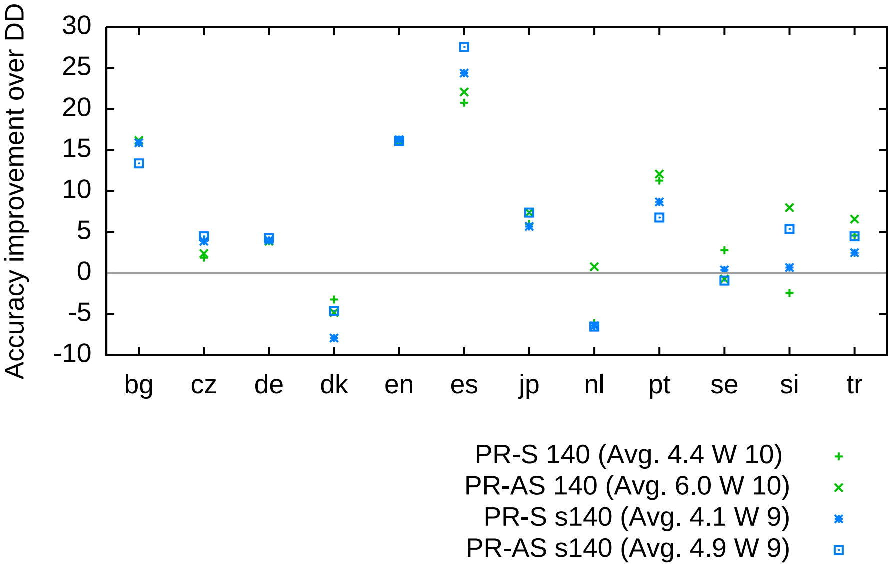
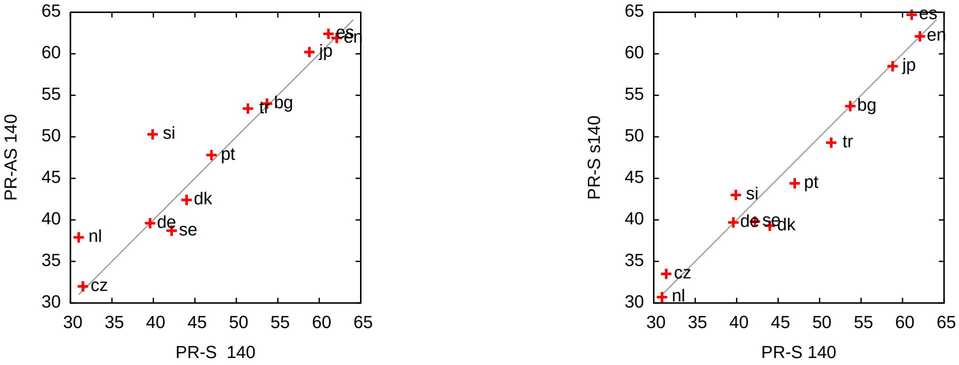


Figure 9: Difference in accuracy between PR training with the different constraints and DD for the DMV model across the 12 languages. Avg. - Average improvement over DD. W - Number of languages better than DD.



(a) PR-S versus PR-AS

(b) No scaling versus scaling

Figure 10: Comparing the different sparsity constraints for the DMV model over twelve different languages. Left: PR-S vs PR-AS. Right: PR-S without scaling vs PR-S with scaling.

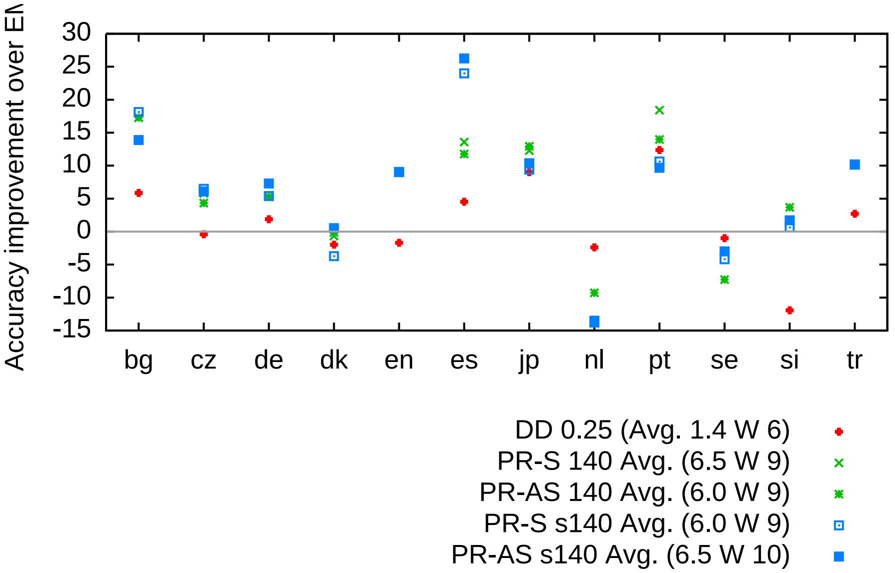


Figure 11: Difference in accuracy between the sparsity inducing training methods and EM training for the E-DMV model with the different training method across the 12 languages. Avg. - Average improvement over DD. W - Number of languages better than EM.

	Bg	Cz	De	Dk	En	Es	Jp	Nl	Pt	Se	Si	Tr
	DMV Model											
EM	37.8	29.6	35.7	**47.2**	45.8	40.3	52.8	37.1	35.7	39.4	42.3	46.8
DD 0.25	39.3	30.0	38.6	43.1	46.4	47.5	57.8	35.1	38.7	40.2	48.8	43.8
PR-S 140	53.7	31.5	39.6	44.0	62.1	61.1	58.8	31.0	47.0	**42.2**	39.9	51.4
PR-AS 140	**54.0**	32.0	39.6	42.4	61.9	62.4	**60.2**	**37.9**	**47.8**	38.7	**50.3**	**53.4**
PR-S s140	53.7	33.5	39.7	39.3	**62.1**	64.7	58.5	30.7	44.4	39.8	43.0	49.3
PR-AS s140	51.2	**34.1**	**40.0**	42.6	61.9	**67.9**	**60.2**	30.6	42.5	38.5	47.7	51.3
	Extended Model											
EM-(3,3)	41.7	48.9	40.1	46.4	55.3	44.3	48.5	**47.5**	35.9	**48.6**	47.5	46.2
DD-(4,4) 0.1	47.6	48.5	42.0	44.4	53.6	48.9	57.6	45.2	48.3	47.6	35.6	48.9
PR-S(3,3) 140	59.0	54.7	**47.4**	45.8	64.3	57.9	60.8	33.9	**54.3**	45.6	49.1	56.3
PR-AS(4,4) 120	59.0	53.2	45.4	46.3	**64.4**	56.1	**61.5**	38.3	49.8	41.3	**51.2**	**56.4**
PR-S(3,3) s140	**59.9**	**55.4**	45.5	42.7	64.3	68.3	57.9	34.0	46.5	44.4	48.1	**56.4**
PR-AS (4,4) s120	55.6	55.0	**47.4**	**46.9**	**64.4**	**70.6**	58.9	33.8	45.6	45.6	49.2	56.3
	Scaled Strengths											
σ120	89	445	246	37	120	9	138	138	48	76	11	58
σ140	103	519	287	43	140	10	161	161	56	89	13	68

Table 5: Attachment accuracy results. For each method we tested both the basic DMV and the E-DMV. The parameters used where the best parameters found for English. For the extended model the child-valency and stop-valency used are indicated in parentheses. **EM**: The EM algorithm. **DD**: Discounting Dirichlet prior. **PR-S**: Our method using the symmetric version of the constraints with strength parameter σ. **PR-S-s**: The same method but strength parameter scaled proportional to the number of tokens in the train set for each language. **PR-AS / PR-AS-s**: Our method with the asymmetric constraints, without and with scaling of the strength parameter. σ: The scaled weights for each corpus for the different values of the strength parameter used for English. Bold indicates the best method for each learning and model type.

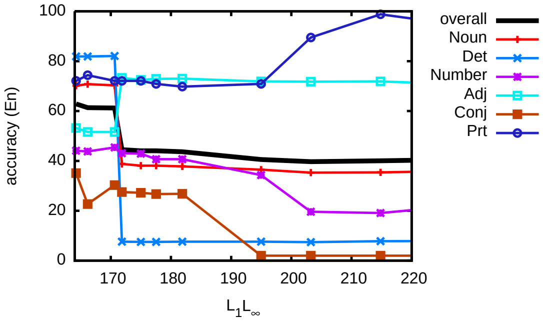


Figure 12: The accuracy overall and for different POS tag types in the English corpus as a function of ℓ_1/ℓ_∞ as we vary the constraint strength. EM has ℓ_1/ℓ_∞ of 431.17.

6. Analysis

6.1 Instability

In our experiments, we saw that the model was somewhat unstable with respect to the regularization strength. Figure 12 shows the accuracies on the English corpus broken down by POS tag category.

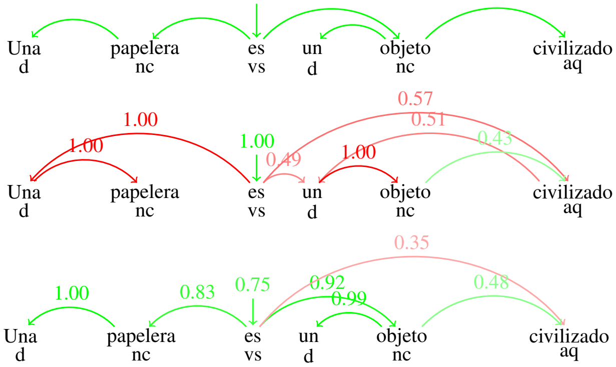


Figure 13: Posterior edge probabilities for an example sentence from the Spanish test corpus. **Top** is Gold, **middle** is EM, and **bottom** is PR.

The plot shows that sharp changes in overall accuracy are in fact caused by even sharper changes in the attachment accuracies of the tag categories. This should not be surprising, given that whether using EM or PR, the objective has many local maxima with deep valleys between them. The problem continues to be very underspecified, and without knowing the "true" sparsity pattern of a language, we can only achieve limited parsing accuracy.

6.2 Comparison of EM, PR, and DD Errors

One common EM error that PR fixes in many languages is the directionality of the noun-determiner relation. Figure 13 shows an example of a Spanish sentence where PR significantly outperforms standard EM because of this fixed relation. As is evidenced in this case, EM frequently assigns a determiner as the parent of a noun, instead of the reverse. PR tends not to make this error. One explanation for this improvement is that it is a result of the fact that nouns can sometimes appear without determiners. For example, consider the sentence "Lleva tiempo entenderlos" (translation: "It takes time to understand") with tags "main-verb common-noun main-verb". In this situation EM must assign the noun to a parent that is not a determiner. In contrast, when PR sees that sometimes nouns can appear without determiners but that the opposite situation does not occur, it shifts the model parameters to make nouns the parent of determiners instead of the reverse, since then it does not have to pay the cost of assigning a parent with a new tag to cover each noun that doesn't come with a determiner.

Table 6 contrasts the most frequent types of errors EM, DD, and PR make on several test sets where PR does well. The "acc" column is accuracy and the "errs" column is the absolute number of errors of the key type. Accuracy for the key "parent POS truth/guess → child POS" is computed as a function of the true relation. So, if the key is $p_t/p_g \rightarrow c$, then accuracy is:

$$\text{acc} = \frac{\text{\# of } p_t \rightarrow c \text{ in Viterbi parses}}{\text{\# of } p_t \rightarrow c \text{ in gold parses}}. \tag{21}$$

In the following subsections we provide some analysis of the results from Table 6.

	EM			DD			PR		
	key	**acc**	**errs**	**key**	**acc**	**errs**	**key**	**acc**	**errs**
es	sp/d → nc	0.0	7	sp/d → nc	0.0	7	vm/<root> → vm	0.0	5
	nc/sp → d	0.0	6	nc/sp → d	0.0	6	<root>/vm → vm	0.0	4
	vm/d → nc	0.0	5	vm/<root> → vm	0.0	6	<root>/vm → vs	0.0	3
	vs/d → nc	0.0	4	nc/vm → d	0.0	6	rg/vm → rg	0.0	2
	vm/<root> → vm	0.0	4	vm/d → nc	0.0	5	aq/aq → cc	0.0	2
	nc/vm → d	0.0	4	<root>/vm → vm	0.0	4	nc/cc → aq	0.0	2
	aq/<root> → cc	0.0	3	vs/d → nc	0.0	4	vs/<root> → vm	0.0	2
	<root>/vm → vm	0.0	3	vm/p → rn	0.0	3	aq/nc → aq	0.0	2
	vm/p → rn	0.0	3	nc/vs → d	0.0	3	vm/vm → sp	75.0	2
	nc/vs → d	0.0	3	nc/<root> → d	0.0	3	vs/vm → cs	0.0	2
	vm/nc → sp	0.0	3	vm/nc → sp	0.0	3	vm/nc → sp	0.0	2
	vm/cs → vs	0.0	2	<root>/rg → vm	0.0	2	aq/cc → aq	0.0	1
	vm/d → p	0.0	2	nc/p → d	0.0	2	nc/vs → aq	0.0	1
	nc/aq → d	0.0	2	<root>/d → nc	0.0	2	<root>/aq → nc	0.0	1
	<root>/vm → vs	0.0	2	aq/cc → aq	0.0	2	vm/vm → cc	50.0	1
bg	<root>/R → V	0.0	65	N/V → R	0.0	53	N/V → R	0.0	56
	N/<root> → R	0.0	37	V/R → N	0.0	47	V/R → N	0.0	46
	V/<root> → R	0.0	29	<root>/C → V	0.0	26	T/V → V	0.0	26
	V/R → R	0.0	24	V/R → R	0.0	25	V/R → R	0.0	25
	N/M → N	0.0	20	T/V → V	0.0	23	V/V → T	42.4	19
	V/V → T	40.6	19	N/M → N	0.0	20	N/N → N	73.4	17
	<root>/C → V	0.0	18	V/V → T	42.4	19	V/V → N	84.8	14
	V/<root> → C	0.0	17	V/<root> → C	0.0	17	V/V → C	30.0	14
	T/V → N	0.0	17	N/<root> → C	0.0	15	T/V → N	0.0	13
	N/<root> → C	0.0	16	R/N → N	0.0	14	<root>/V → T	0.0	11
	V/R → N	0.0	16	T/V → N	0.0	13	N/V → V	0.0	10
	<root>/T → V	0.0	15	V/N → N	0.0	11	T/V → P	0.0	10
	N/V → R	0.0	15	N/R → N	0.0	10	N/N → M	66.7	10
	T/<root> → V	0.0	12	V/V → N	87.3	10	V/N → N	0.0	10
	R/N → N	0.0	12	N/V → V	0.0	10	<root>/V → V	0.0	9
pt	n/prp → art	0.0	39	n/prp → art	0.0	37	prp/v-fin → n	0.0	32
	v/art → n	0.0	31	v/art → n	0.0	32	n/prp → art	0.0	27
	prp/art → n	0.0	24	prp/art → n	0.0	27	v/n → prp	0.0	22
	n/v-fin → prp	0.0	18	n/v-fin → art	0.0	21	n/n → prp	0.0	20
	n/v-fin → art	0.0	17	v/v-fin → prp	72.5	11	v/prp → n	0.0	18
	v/pron-det → n	0.0	12	n/v-fin → prp	0.0	10	prp/v-fin → prop	0.0	11
	v/v-fin → prp	69.4	11	prop/prp → art	0.0	8	prp/prp → n	0.0	11
	v/prp → v	0.0	11	v/v-fin → adv	68.0	8	v/v-fin → adv	64.0	9
	prp/pron-det → n	0.0	10	prp/art → prop	0.0	7	prop/prp → art	0.0	8
	v/prp → prp	0.0	9	v/prp → v	0.0	7	v/v-fin → n	81.0	8
	prop/prp → art	0.0	8	v/prp → n	0.0	7	v/prop → prp	0.0	8
	n/v-fin → pron	0.0	8	<root>/conj-c → v	0.0	5	n/prop → prp	0.0	8
	n/prp → pron	0.0	8	v/<root> → v	0.0	5	v/v-fin → prp	58.8	7
	n/<root> → prp	0.0	8	v/art → prop	0.0	5	v/prp → v	0.0	7
	prp/art → prop	0.0	7	n/<root> → prp	0.0	5	<root>/prp → n	0.0	6
en	VB/DT → NN	0.0	129	VB/DT → NN	0.0	133	NN/NNP → NN	54.2	76
	NN/NNP → NN	60.1	65	NN/NNP → NN	54.7	78	IN/NN → NN	0.0	37
	NN/VBZ → DT	0.0	52	NN/IN → DT	0.0	56	MD/<root> → VB	0.0	25
	NN/IN → DT	0.0	47	NN/VBZ → DT	0.0	52	<root>/VB → MD	0.0	25
	IN/DT → NN	0.0	46	IN/DT → NN	0.0	46	IN/NNS → NN	0.0	24
	NN/VBD → DT	0.0	41	NN/VBD → DT	0.0	35	VB/NN → IN	0.0	21
	VB/TO → VB	0.0	19	VB/TO → VB	0.0	19	NN/NN → DT	86.5	21
	NN/VBP → DT	0.0	19	NN/VBP → DT	0.0	18	VB/DT → IN	0.0	20
	<root>/CD → NN	0.0	14	NN/NN → JJ	78.9	16	IN/VBD → NN	0.0	18
	NN/NN → JJ	81.1	14	VB/IN → JJ	0.0	12	NN/NN → JJ	79.2	16
	NN/VB → DT	0.0	14	VB/PRP$ → NN	0.0	12	IN/VBZ → NN	0.0	15
	NN/CD → CD	0.0	13	<root>/CD → NN	0.0	12	IN/VBP → NN	0.0	13
	VB/PRP$ → NN	0.0	12	NN/VB → DT	0.0	12	VB/VB → RB	18.8	13
	VB/DT → RB	0.0	11	NN/<root> → CD	0.0	11	NN/<root> → NN	0.0	11
	VB/<root> → VB	0.0	10	VB/NNS → RB	0.0	11	VB/NNS → NN	0.0	11

Table 6: Top 15 mistakes by parent POS truth/guess → child POS for English and the three languages where PR makes the greatest gains over EM with the E-DMV.

6.3 English Corrections

Considering English first, there are several notable differences between EM and PR errors. Similar to the example for Spanish, the direction of the noun-determiner relation is corrected by PR. This is reflected by the VB/DT → NN key, the NN/VBZ → DT key, the NN/IN → DT key, the IN/DT → NN key, the NN/VBD → DT key, the NN/VBP → DT key, and the NN/VB → DT key, which for EM and DD have accuracy 0. PR corrects these errors.

A second correction PR makes is reflected in the VB/TO → VB key. One explanation for the reason PR is able to correctly identify VBs as the parents of other VBs instead of mistakenly making TO the parent of VBs is that "VB CC VB" is a frequently occurring sequence. For example, "build and hold" and "panic and bail" are two instances of the "VB CC VB" pattern from the test corpus. Presented with such scenarios, where there is no TO present to be the parent of VB, PR chooses the first VB as the parent of the second. It maintains this preference for making the first VB a parent of the second when encountered with "VB TO VB" sequences, such as "used to eliminate", because it would have to pay an additional penalty to make TO the parent of the second VB. In this manner, PR corrects the VB/TO → VB key error of EM and DD.

A third correction PR makes is reflected in the <root>/CD → NN key. This correction is similar to the noun-determiner correction: CD and NN often co-occur, but while CD almost never appears without NN, NN frequently appears without CD. Thus, if PR chose CD as parent of NN, it would have to pay an additional penalty to select another parent for NN in sentences where no CDs exist. Thus, PR is able to recognize that CD is not usually a good parent for NN. Again, EM and DD have 0 accuracy for this key.

There are a couple of errors common to EM, DD, and PR. These correspond to the NN/NN → JJ key and the NN/NNP → NN key. These are notoriously difficult relations to get right, especially for an unlexicalized model that also has no notion of the surface lengths of relations. We predict that combining PR with a model such as the lexicalized DMV of Headden III et al. (2009), or applying the structural annealing technique of Smith and Eisner (2006), could greatly reduce these types of errors. These changes could also help reduce some of the other main errors PR makes, such as the ones corresponding to the keys NN/NN → DT and VB/VB → RB.

Even after all these improvements, there would likely persist at least one type of English error that would be hard to fix: the domination of modals by verbs. By convention, modals dominate verbs in English dependency parses. This is a relatively arbitrary choice, as there are linguistically sound arguments to be made for either dominating the other. In fact, in some of the other languages we work with the annotation convention is the reverse of what it is in English. Thus, for now we merely note that the keys MD/<root> → VB and <root>/VB → MD account for a large portion of the English errors with PR.

6.4 Bulgarian Corrections

Moving beyond English, let's consider Bulgarian. We might expect qualitatively different results for Bulgarian for two reasons. First, the language is not in the same family as English. Second, the Bulgarian corpus employs far fewer POS tags.

One large correction PR makes with respect to EM and DD corresponds to the key N/M → N. The tag M stands for "numeral" in the Bulgarian corpus, so this correction is similar to the English correction involving the tag CD. Another substantial correction PR makes with respect to EM and DD corresponds to the key <root>/C → V. The tag C stands for "conjunction" in the Bulgarian corpus, so this correction means the model is realizing verbs should usually be sentence roots rather than children of conjunctions. Following the usual line of reasoning as to why PR achieves this correction, we note that sentences with verbs but no conjunctions are very common, so if PR chose C as the parent of V, it would have to pay a penalty to give V a different parent in such sentences. The same reasoning explains why PR doesn't see the V/<root> → C errors or the N/<root> → C errors that EM and DD do.

Although PR is able to make great improvements for Bulgarian parsing, it is clearly crippled by the small number of POS tags. EM, DD, and PR all make substantial errors in deciding which verb to use as the parent of a particle (see key V/V → T), and many of the main remaining errors for PR are caused by similar symmetries (see keys N/N → N, V/V → N, V/V → C, N/N → M, and <root>/V → V). As mentioned in the analysis of English, lexicalization or incorporation of a notion of surface length of relations might help alleviate these problems.

Corrections PR makes in the other languages can be analyzed using the same type of reasoning as we have applied to analysis of English and Bulgarian. We thus leave more extensive interpretation of Table 6 to the reader.

7. Conclusion

In this paper we presented a new method for unsupervised learning of dependency parsers. In contrast with previous approaches that impose a sparsity bias on the model parameters using discounting Dirichlet distributions, we impose a sparsity bias on the model posteriors. We do so by using the posterior regularization (PR) framework (Graça et al., 2007) with constraints that favor posterior distributions that have a small number of unique parent-child relations. We propose two such constraints: a symmetric constraint similar in spirit to the sparsity constraint applied to part-of-speech (POS) induction by Graça et al. (2009), and an asymmetric version of the same constraint that more directly tries to minimize the number of different parent-child types instead of different parent-child occurrences. On English our approach consistently outperforms the standard EM algorithm and the approach of training in a Bayesian setting where a discounting Dirichlet prior is used. Moreover, we perform an extensive comparison with previous published work and show that our learning approach achieves state-of-the-art results. We compare our approach on 11 additional languages, which as far as we know is the most extensive comparison made for a dependency parser. We report significant improvements over the competing learning approaches. The new approach beats EM training for 10 out of 12 languages with an average improvement of 6.5%. It also beats the Bayesian learning approach for 9 out of 12 languages with average improvement of 4% to 5%.

One significant problem we encountered was picking the different parameters for the model in an unsupervised way, for which we found no good principled solution that worked for all languages. The likelihood on heldout development sets does not seem to be a reliable proxy for the model quality. Besides which, choosing the complexity parameters for the E-DMV model cannot be done based on likelihood since the models are different. As future work we intend to investigate additional unsupervised measures for quality of dependency parses, following the recent work of Reichart and Rappoport (2009). Even in the absence of a good unsupervised measure of model quality, a better formula for transferring the regularization strength parameter from one language to another is also needed. The regularization strength is strongly dependent on the corpus, both on the number of parent-child pairs being constrained as well as on the number of tokens for each parent and child.

Our experiments approximated this dependence by scaling the best English regularization strength by the number of tokens in other corpora, but this is not ideal.

With respect to model initialization, the K&M initialization is highly biased to the simple DMV model, and both RandomP initialization and the initialization approaches proposed by Spitkovsky et al. (2010) can significantly boost the performance of the model. We wish to initialize our models with the approaches proposed by Spitkovsky et al. (2010), since besides producing better results, those approaches are deterministic and reduce the number of parameters that need to be tuned. Following the spirit of these initialization approaches, we also propose that some success might be had by initializing the simple DMV training it, and then using its learned parameters to initialize more complex models (E-DMV models with larger valence values).

Regarding the sparsity constraints, we note that the versions we are using do not take into account some possibly important information, such as the directionality of the edge. Moreover, the same strength is currently used for the root probabilities and for the parent-child probabilities. Also, we could extend the constraints to work directly on word types rather than on POS tags, since there is a lot of information lost by discarding the particular words. For instance, Headden III et al. (2009) achieve significant improvements by conditioning the edge probabilities on the parent word together with the parent POS. Additionally, we could explore other constraints to encourage locality by preferring short dependency edges as suggested by the SA work of Smith (2006).

Finally, we would like in the future to move to fully unsupervised learning of grammar. That is, we would like to use POS tags induced in an unsupervised manner, instead of assuming gold POS tags, and see how robust our method is under these conditions. Recent studies show that the quality of the DMV model degrades significantly when the induced POS tags are used Headden III et al. (2008). It would be interesting to see if our model is more robust to the quality of the provided tags. Further, it would be even more interesting to see how our method performs if we applied it to aid in the more complex task of joint induction of POS tags and dependency parses.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. Floresta Sinta(c)tica: a treebank for Portuguese. In *Proc. LREC*, 2002.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *Proc. UAI*, 2009.

D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1995.

A. Bohomovà, J. Hajic, E. Hajicova, and B. Hladka. The prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, 2001.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, 2002.

M. Civit and M.A. Martí. Building cast3lb: A Spanish Treebank. *Research on Language & Computation*, 2004.

S.B. Cohen and N.A. Smith. The shared logistic normal distribution for grammar induction. In *Proc. NAACL*, 2009.

S.B. Cohen, K. Gimpel, and N.A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proc. NIPS*, 2008.

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

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. Towards a Slovene dependency treebank. In *Proc. LREC*, 2006.

J. Finkel, T. Grenager, and C. Manning. The infinite tree. In *Proc. ACL*, 2007.

K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, to appear 2010.

J. Graça, K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In *Proc. NIPS*, 2007.

J. Graça, K. Ganchev, B. Taskar, and F. Pereira. Posterior sparsity vs parameter sparsity. In *Proc. NIPS*, 2009.

J. Graça, K. Ganchev, and B. Taskar. Learning tractable word alignment models with complex constraints. *Computational Linguistics*, to appear 2010.

W. Headden III, D. McClosky, and E. Charniak. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proc. CoNLL*, 2008.

W.P. Headden III, M. Johnson, and D. McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. NAACL*, 2009.

M. Johnson, T.L. Griffiths, and S. Goldwater. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Proc. NIPS*, 2007.

Y. Kawata and J. Bartels. Stylebook for the Japanese Treebank in VERBMOBIL. Technical report, Eberhard-Karls-Universitat Tubingen, 2000.

D. Klein and C. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. ACL*, 2004.

M.T. Kromann, L. Mikkelsen, and S.K. Lynge. Danish Dependency Treebank. In *Proc. TLT2003*, 2003.

K. Kurihara and T. Sato. An application of the variational Bayesian approach to probabilistic context-free grammars. In *IJC-NLP Workshop: Beyond Shallow Analyses*, 2004.

P. Liang, S. Petrov, M.I. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Proc. EMNLP*, 2007.

P. Liang, M.I. Jordan, and D. Klein. Learning from measurements in exponential families. In *Proc. ICML*, 2009.

G. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proc. ICML*, 2007.

G. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *Proc. ACL*, 2008.

M. Marcus, M. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

D. McClosky. Modeling valence effects in unsupervised grammar induction. Technical report, CS-09-01, Brown University, 2008.

R. Neal and G. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

J. Nilsson and J. Hall, J.and Nivre. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. *NODALIDA Special Session on Treebanks*, 2005.

K. Oflazer, B. Say, D.Z. Hakkani-Tür, and G. Tür. Building a Turkish treebank. *Treebanks: Building and Using Parsed Corpora*, 2003.

R. Reichart and A. Rappoport. Automatic selection of high quality parses created by a fully unsupervised parser. In *Proc.CoNLL*, 2009.

K. Simov, P. Osenova, M. Slavcheva, S. Kolkovska, E. Balabanova, D. Doikoff, K. Ivanova, A. Simov, E. Simov, and M. Kouylekov. Building a linguistically interpreted corpus of bulgarian: the bultreebank. In *Proc. LREC*, 2002.

N. Smith. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text.* PhD thesis, Johns Hopkins University, 2006.

N. Smith and J. Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. IJC-AI Workshop: Grammatical Inference Applications*, 2005a.

N. Smith and J. Eisner. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. IJC-AI Workshop: Grammatical Inference Applications*, 2005b.

N. Smith and J. Eisner. Annealing structural bias in multilingual weighted grammar induction. In *Proc. ACL*, 2006.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. From Baby Steps to Leapfrog: How “Less is More” in unsupervised dependency parsing. In *Proc. of NAACL-HLT*, 2010.

P. Tseng. An analysis of the EM algorithm and entropy-like proximal point methods. *Mathematics of Operations Research*, 29(1):27–44, 2004.

L. Van der Beek, G. Bouma, R. Malouf, and G. Van Noord. The Alpino dependency treebank. *Language and Computers*, 2002.