Journal of Machine Learning Research 12 (2011) 455-490 Submitted 5/10; Revised 1/11; Published 2/11

Posterior Sparsity in Unsupervised Dependency Parsing

Jennifer Gillenwater JENGI@CIS.UPENN.EDU
Kuzman Ganchev KUZMAN@CIS.UPENN.EDU
João Graça GRACA@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Levine 302, 3330 Walnut St
Philadelphia, PA 19104, USA

Fernando Pereira PEREIRA@GOOGLE.COM
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA

Ben Taskar TASKAR@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Levine 302, 3330 Walnut St
Philadelphia, PA 19104, USA

Editor: Mark Johnson

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 attachment accuracy over the standard expectation maximization (EM) baseline, with an average accuracy improvement of 6.5%, outperforming EM by at least 1% for 9 out of 12 languages. Furthermore, the new method outperforms models based on standard Bayesian sparsity-inducing parameter priors with an average improvement of 5% and positive gains of at least 1% for 9 out of 12 languages. 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 expect 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 mostly focused on achieving 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 (non-hierarchical) 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) investigated a (shared) logistic normal prior, and Headden III et al. (2009) used a backoff scheme.

Our experiments (Section 6) 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. Our experiments validate that this translates into accuracy improvements. In all except one of the 60 model settings we try for English, we observe higher accuracy than with the *best* setting for a parameter prior baseline. In our multi-lingual experiments, we similarly observe an average absolute accuracy gain of 5%.

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; Ganchev et al., 2010). Specifically, to implement PR we augment the maximum likelihood objective of the generative dependency model with a term that penalizes distributions over head-dependent pairs 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 the same head tag p for any other occurrence of c is free. While Ravi et al. (2010) also attempt a direct minimization of tag pairs for a supertagging application, they do so with a two-stage integer program that is applied after likelihood maximization is complete.

The remainder of this paper is organized as follows. Section 2 reviews the generative model for dependency parsing. Section 3 illustrates why the expectation-maximization learning method is insufficient and motivates sparse posteriors. Section 4 describes learning with PR constraints and how to encode posterior sparsity under the PR framework. Section 5 summarizes previous approaches that we compare to in our experiments, focusing in particular on attempts to induce sparsity via a parameter prior. Section 6 describes the results of dependency parsing experiments across 12 languages and against recent published state-of-the-art results for the English language. Section 7 analyzes these results, explaining why PR manages to learn where other methods fail, and Section 8 concludes. The model and all the code required to reproduce the experiments are available online at `code.google.com/p/pr-toolkit`, version 2010.11.

2. Parsing Model

The models we consider are based on the dependency model with valence (DMV) of Klein and Manning (2004). 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.

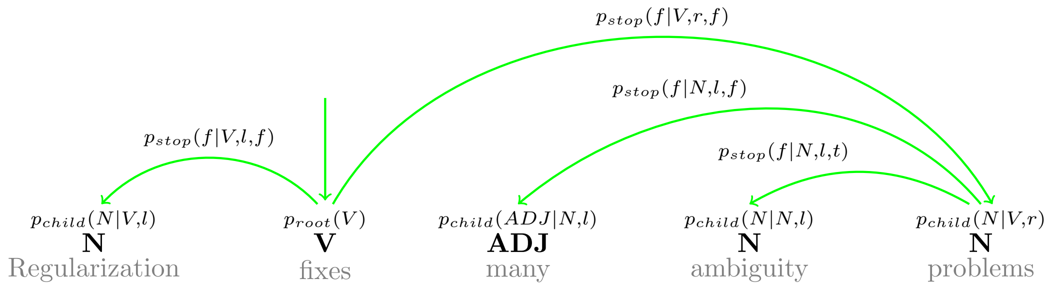


Figure 1: Example of a dependency tree with DMV probabilities. Right-dependents of a head are denoted by r, left-dependents by l. The letters t and f denote 'true' and 'false.' For example, in $p_{stop}(f \mid V,r,f)$ the f to the left of the conditioning bar indicates that the model has decided *not*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

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. In this section we briefly describe the basic DMV model. Description of the extended models is deferred until the experiments section.

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 one 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 image of the right-dependent process. Once the root has generated all of its dependents, the dependents generate their own dependents in the same manner.

We follow the convention 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 is exemplified in Figure 1, where the leftmost dependent of the final N is generated before the other left dependent. This convention has no effect on the final probability of a parse tree under the basic DMV. However, as we will note in the experiments section, it does affect dependency tree probabilities in the extended model.

3. Learning with EM

The baseline for evaluating our sparse learning methods is the expectation maximization (EM) algorithm (Dempster et al., 1977). Before the empirical comparison in Section 6, in we introduce here some notation and review the EM algorithm. In what follows, we denote the entire unlabeled corpus by $\mathbf{X} = \{\mathbf{x}^1, \dots, \mathbf{x}^n\}$, and a set of corresponding parses for each corpus sentence by $\mathbf{Y} = \{\mathbf{y}^1, \dots, \mathbf{y}^n\}$.

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 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 of the basic EM algorithm. 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 $\mathcal{L}(\theta)$:

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

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

$$\begin{aligned} F(q,\theta) &= \sum_{\mathbf{Y}} q(\mathbf{Y}) \log(p_\theta(\mathbf{X}) p_\theta(\mathbf{Y} \mid \mathbf{X})) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log q(\mathbf{Y}) \\ &= \mathcal{L}(\theta) - \sum_{\mathbf{Y}} q(\mathbf{Y}) \log \frac{q(\mathbf{Y})}{p_\theta(\mathbf{Y} \mid \mathbf{X})} \qquad (1) \\ &= \mathcal{L}(\theta) - \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y} \mid \mathbf{X})). \end{aligned}$$

Based on this formulation, 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:

$$\begin{aligned} \mathbf{E} : q^{t+1} &= \arg\max_q F(q, \theta^t) = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y} \mid \mathbf{X})), \\ \mathbf{M} : \theta^{t+1} &= \arg\max_\theta F(q^{t+1}, \theta) = \arg\max_\theta \mathbf{E}_{q^{t+1}} \left[\log p_\theta(\mathbf{X}, \mathbf{Y})\right]. \qquad (2) \end{aligned}$$

Note that the E-step just sets $q^{t+1}(\mathbf{Y}) = p_{\theta^t}(\mathbf{Y}|\mathbf{X})$, since it performs 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 posterior regularization reduces the mismatch much more successfully than approaches based on parameter priors.

4. Learning with Sparse Posteriors

We stated in the introduction that posterior regularization makes gains over baseline methods such as EM by inducing sparsity in the posteriors. Before discussing how to learn a model with sparse posteriors, we wish to further motivate the idea. The main intuition behind our method is that a

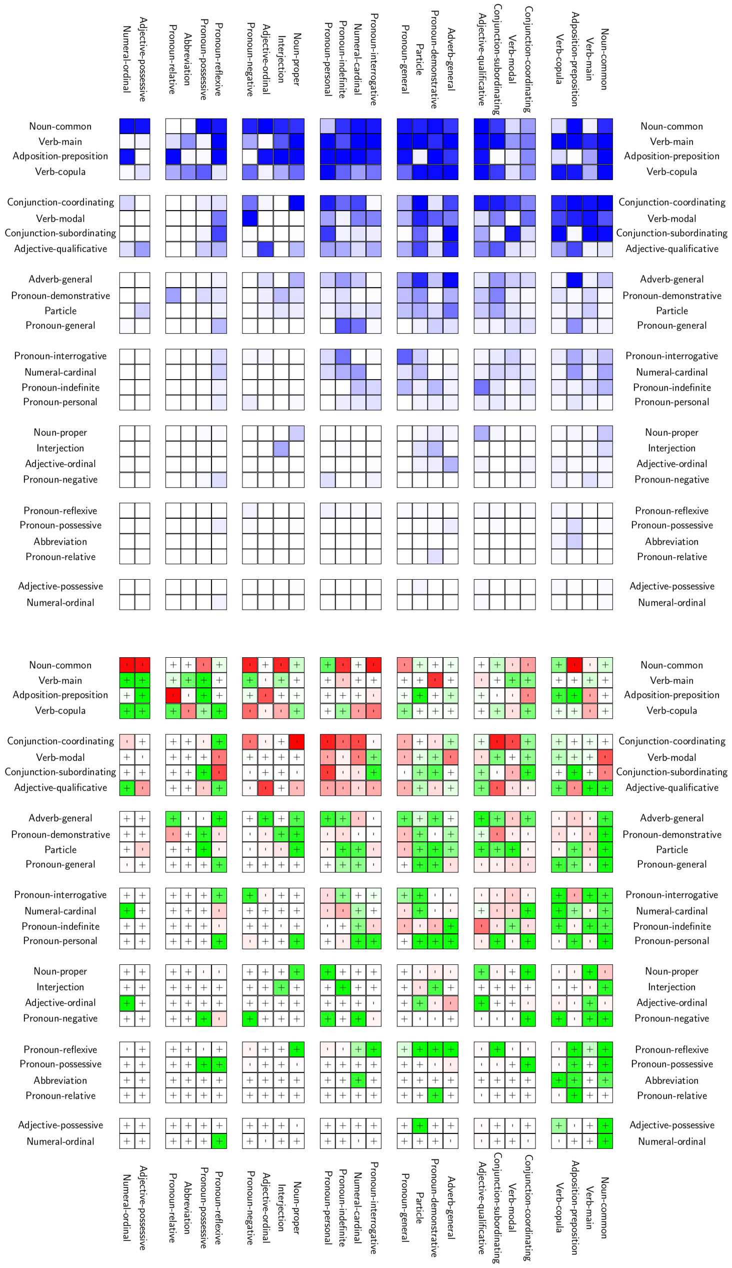


Figure 2: Comparison of posteriors for a maximum likelihood DMV and an EM-trained DMV for Slovene. Each square represents a parent-child pair. Parent tags are listed down, child tags across. Parent tags are sorted top-to-bottom in descending order by the number of unique child tags they take. **Top**: Using maximum likelihood parameter settings (supervised). The saturation of a square with parent p and child c is determined by the max value of the posterior probability of type c having parent p observed in the entire English training corpus (Marcus et al., 1993). More saturated blue indicates higher probability. **Bottom**: Using EM parameter settings. Green ("+") indicates EM posteriors are too high, red ("-") too low. More saturation indicates more deviation. There are significantly more green ("+") squares than red ("-"), indicating EM does not learn a sparse enough model.

useful grammar should only allow a relatively small subset of all possible parent-child relations. If we were asked to parse the tag sequence `DT ADJ N V`, the dependency tree with V as root, N as its child, and the remaining DT and ADJ as N's children is almost forced on us. Yet, if the English grammar allowed all possible parent-child relations, you would have had to consider 30 different (projective) parse trees before selecting the correct one. Knowledge of unlikely relations simplifies parsing for us. Thus, in this work we attempt to limit grammar ambiguity by inducing a grammar that allows only a sparse set of possible dependency relation types.

Empirical evidence that good grammars have sparse coverage of the possible parent-child relations can be seen in Figure 2. The grid corresponding to supervised parameter settings has many white squares, which illustrates that many parent-child relations should have zero posterior. Notice also that while some parent tags can take many different child tags, some parent tags can take just a few child tags, and some tags cannot be parents; the number of allowed child tags spans a wide range. These empirical properties are not captured by previous attempts to achieve model sparsity with hierarchical Bayesian models, which push each *each* parent tag to allow only a few child tags. Instead, the modeling framework should simply favor models with high overall ratio of white squares to blue squares.

The foregoing argument leads us to seek learning methods that will penalize learned distributions $p_\theta(\mathbf{Y}|\mathbf{X})$ that predict a large number of distinct dependency types. In the next section, we discuss different ways of counting dependency types, corresponding to slightly different measures of ambiguity. In Section 4.3, we will explain how to use those measures as mixed-norm penalties on distributions over dependency trees.

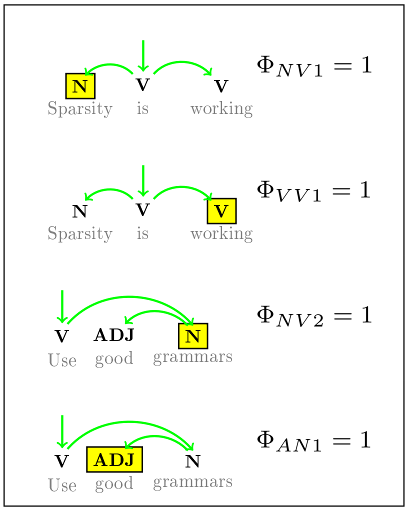
We will then discuss how to apply the posterior regularization (PR) framework (Graça et al., 2007; Ganchev et al., 2010) to achieve the desired sparsity in grammar induction. The approach, reviewed in Section 4.2, is closely related to generalized expectation constraints (Mann and McCallum, 2007, 2008; 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 ambiguity measures described below. For a more complete analysis of PR and its application to a variety of NLP tasks, we refer the reader to Ganchev et al. (2010).

4.1 Measures of Ambiguity

We now describe precisely how to count dependency types, which will allow us to specify different kinds of dependency 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:

$$\sum_{cp} \max_i \phi_{cpi}(\mathbf{X},\mathbf{Y}), \tag{3}$$

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. See the top sentence in Figure 4 for an example of this; there the noun child in the POS sequence N



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

Figure 3: The ℓ_1/ℓ_∞ ambiguity measure for a toy example with gold parse trees. 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. **Right**: Computation of the grammar ambiguity measure, which is 3 in this case. The same result can also be obtained using ϕ_{cpij} instead.

V V is considered, and the probabilities of each of its possible parents are summed into one factor, Φ_{NV1}, since the parents are both of the same type (V). We use PR-AS, asymmetric PR, to refer to PR training with constraints based on this ambiguity measure.

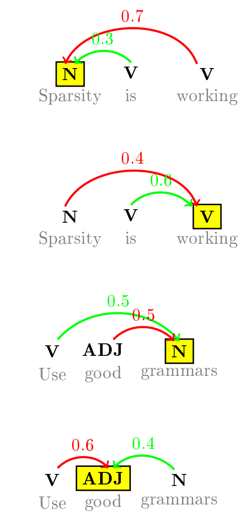
Instead of counting pairs of a child token and a parent type, we could instead have counted pairs of a child token and a parent token by letting p range over all *tokens* rather than *types*. In that case, each potential dependency would correspond to a different indicator ϕ_{cpij}, and the penalty would be symmetric with respect to parents and children. We use PR-S, symmetric PR, to refer to PR training with constraints based on this measure. The number of unique dependency types in this case is given by:

$$\sum_{cp} \max_{i,j} \phi_{cpij}(\mathbf{X}, \mathbf{Y}).$$

On actual dependency trees, where each child has a unique parent, PR-AS and PR-S always yield the same value. However, the values may be different when working with distributions over edge types instead, as exemplified in Figure 4. Both PR-AS and PR-S perform very well. One approach is not clearly better than the other when compared across the twelve languages, so we report results for both versions in the results section.

In addition to PR-AS and PR-S, there is in fact a third way of counting—another asymmetric method. For PR-AS all parent tokens are collapsed, but we could also consider the case where all child tokens are collapsed. Then the number of unique dependency types would be:

$$\sum_{cp} \max_{j} \phi_{cpj}(\mathbf{X}, \mathbf{Y}).$$



PR-AS	PR-S
$\Phi_{NV1} = 1$	$\Phi_{NV11} = 0.3$ $\Phi_{NV12} = 0.7$
$\Phi_{VN2} = 0.4$ $\Phi_{VV2} = 0.6$	$\Phi_{VN21} = 0.4$ $\Phi_{VV21} = 0.6$
$\Phi_{NA2} = 0.5$ $\Phi_{NV2} = 0.5$	$\Phi_{NA21} = 0.5$ $\Phi_{NV23} = 0.5$
$\Phi_{AV1} = 0.6$ $\Phi_{AN1} = 0.4$	$\Phi_{AV13} = 0.6$ $\Phi_{AN12} = 0.4$

PR-Asymmetric

$$\begin{aligned}
\xi_{\mathbf{NV}} &= \max(\Phi_{NV1}, \Phi_{NV2}) = 1\\
\xi_{VN} &= \max(\Phi_{VN2}) = 0.4\\
\xi_{\mathbf{VV}} &= \max(\Phi_{VV2}) = 0.6\\
\xi_{NA} &= \max(\Phi_{NA2}) = 0.5\\
\xi_{AV} &= \max(\Phi_{AV1}) = 0.6\\
\xi_{\mathbf{AN}} &= \max(\Phi_{AN1}) = 0.4
\end{aligned}$$

$$\sum_{cp} \xi_{cp} = \mathbf{3.5}$$

PR-Symmetric

$$\begin{aligned}
\xi_{\mathbf{NV}} &= \max(\Phi_{NV11}, \Phi_{NV12}, \Phi_{NV23}) = 0.7\\
\xi_{VN} &= \max(\Phi_{VN21}) = 0.4\\
\xi_{\mathbf{VV}} &= \max(\Phi_{VV21}) = 0.6\\
\xi_{NA} &= \max(\Phi_{NA21}) = 0.5\\
\xi_{AV} &= \max(\Phi_{AV13}) = 0.6\\
\xi_{\mathbf{AN}} &= \max(\Phi_{AN12}) = 0.4
\end{aligned}$$

$$\sum_{cp} \xi_{cp} = \mathbf{3.2}$$

Figure 4: The ℓ_1/ℓ_∞ ambiguity measure for a toy example using edge posteriors. Let $\Phi_{cpi} = \mathbf{E}_q[\phi_{cpi}]$, and similarly $\Phi_{cpij} = \mathbf{E}_q[\phi_{cpij}]$. For simplicity we ignore the root $\rightarrow c$ edges here, though in our experiments we incorporate their probabilities also. The two POS tag sequences considered are the same as in 3; we also consider the same four children here for easy comparison. In this unsupervised setting, instead of gold trees we have an example posterior distribution over parents for each child. We illustrate computation of the grammar ambiguity measure for both PR-AS (left), and PR-S(right). Since real grammars tend to have few edge types, it should make sense that the ℓ_1/ℓ_∞ of the set of supervised trees in 3 was smaller.

This type of counting leads however to some unintuitive results. For instance, consider a parse tree consisting of a verb with two noun children. There, $\phi_{NV1} = 2$. This does not correspond to a count of unique parent-child pairs, so it does not serve our ultimate goal as well as PR-AS or PR-S. Hence, we do not experiment with this ambiguity measure in this work.

4.2 Posterior Regularization

Having defined several ambiguity measures, we now step back and describe the general PR framework. After this overview, we will show how to apply this general framework to penalize with respect to the specific ambiguity measures we defined. In general, PR can be seen as a penalty on the standard marginal log-likelihood objective, which we define first as:

$$\textbf{Likelihood objective:}\;\; \mathcal{L}(\theta) = \log p_\theta(\mathbf{X}) + \log p(\theta) \\ = \sum_{\mathbf{x}\in\mathbf{X}} [\log \sum_{\mathbf{y}} p_\theta(\mathbf{x},\mathbf{y})] + \log p(\theta), \tag{4}$$

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. (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:

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

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:

$$\textbf{Penalty term:}\;\; \min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \;+\; \sigma\left|\left|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]\right|\right|_\beta, \tag{5}$$

where σ is the strength of the regularization. As we will see, using an auxiliary distribution q will make the final objective easier to optimize. Ganchev et al. (2010) describe how to compute this penalty term in general, but we will defer that explanation to Section 4.3 when we describe our particular penalty term. The PR framework seeks to maximize:

$$\textbf{PR objective:}\;\; J(\theta) = \mathcal{L}(\theta) - \min_q \left[\mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) \;+\; \sigma\left|\left|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]\right|\right|_\beta\right]. \tag{6}$$

The objective in Equation 6 can be optimized by a variant of the EM algorithm (Dempster et al., 1977) used to optimize the objective in Equation 4.

4.3 ℓ_1/ℓ_∞ Regularization

The previous section gave the penalty version of the PR objective in the general case. We will now show how the ambiguity measures we want to incorporate fit into this framework. Specifically, notice that we can view Equation 3 as a mixed-norm penalty on the features ϕ_{cpi} so that the generic β from Equation 5 becomes ℓ_1/ℓ_∞. 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. To compute the value of the PR objective and also to optimize it, we need to compute the projection:

$$\arg\min_q \mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma\sum_{cp}\max_i \mathbf{E}_q[\phi_{cpi}(\mathbf{X},\mathbf{Y})],$$

which can equivalently be written as:

$$\textbf{Projection}: \min_{q,\xi}\;\; \mathbf{KL}(q(\mathbf{Y}) \parallel p_\theta(\mathbf{Y}|\mathbf{X})) + \sigma\sum_{cp}\xi_{cp} \\ \text{s.t.}\quad \xi_{cp} \geq \mathbf{E}_q[\phi_{cpi}(\mathbf{X},\mathbf{Y})]\;\; \forall c,p,i, \tag{7}$$

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 7 makes the derivation of the dual easier (see Ganchev et al., 2010 for a derivation of the dual in the general case). The dual of the projection problem is a fairly simple convex optimization problem with simplex constraints (scaled by σ):

$$\textbf{Projection dual}: \min_{\lambda \geq 0} \; \log\left(\sum_{\mathbf{Y}} p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{X},\mathbf{Y}))\right) \quad \text{s.t.} \quad \sum_i \lambda_{cpi} \leq \sigma,$$

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 optimal primal solution is related to the dual solution by the equation $q(\mathbf{Y}) \propto p_\theta(\mathbf{Y}|\mathbf{X}) \exp(-\boldsymbol{\lambda} \cdot \phi(\mathbf{X},\mathbf{Y}))$. We solve the dual 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. 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 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.

4.4 Optimization Algorithms

The optimization algorithm for the PR objective uses a minorization-maximization procedure akin to EM. Recall that we defined the PR objective (Equation 6) as:

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

If we further define:

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

then we can express the PR objective in a form very similar to that of the previously introduced lower bound on EM (Equation 1):

$$J(\theta) = \max_q F'(q,\theta).$$

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

$$\mathbf{E}': q^{t+1} = \arg\max_q F'(q,\theta^t) = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}|\mathbf{X})) \; + \; \sigma \left|\left|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]\right|\right|_\beta. \qquad (8)$$

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 2. This scheme is illustrated in Figure 5. The following proposition is adapted from Ganchev et al. (2010), who provide a version for hard constraints.

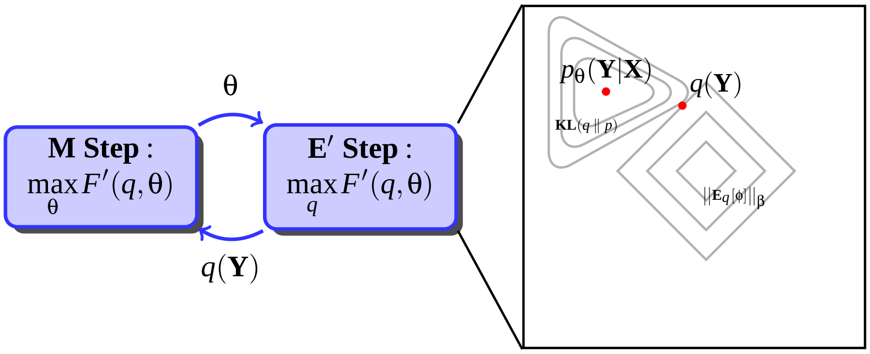


Figure 5: Modified EM for maximizing the PR objective $J(\theta)$ via block-coordinate ascent on lower-bound $F'(q,\theta)$. $\mathbf{E}'$-step minimizes $\mathbf{KL}(q(\mathbf{Y})||p_\theta(\mathbf{Y}|\mathbf{X}))+\sigma\left|\left|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]\right|\right|_\beta$.

Proposition 4.1 *For the modified EM algorithm illustrated in Figure 5, which iterates the* $\mathbf{E}'$*-step (Equation 8) with the normal* **M***-step (Equation 2), 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 $\mathbf{E}'$-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 $F'(q^{t+1},\theta^{t+1}) \geq F'(q^{t+1},\theta^t)$. 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.

We note that optimizing the PR objective does take substantially longer than optimizing likelihood by itself. When optimizing likelihood, we can get the optimal posteriors for an E-step using just one call to the inside-outside algorithm for each sentence. For PR though, the function we are optimizing in the $\mathbf{E}'$-step is a **KL** plus a penalty term, so to find its minimum we have to follow the negative gradient. Each step along the negative gradient requires a call to the inside-outside algorithm—several calls if the initial step size we try does not satisfy the Wolfe conditions. Thus, it might be better to use an optimization schedule where $\mathbf{E}'$-step would not be fully optimized in earlier iterations, perhaps taking just a single step along the negative gradient. Then, in later $\mathbf{E}'$-steps, we could increase the precision of the optimization by taking more gradient descent steps (if they are required to get close to the minimum). Fortunately, in practice we found that, at least for the experiments in this paper, the optimization did not take so long that such a schedule was necessary.

5. Prior Learning Approaches and Model Extensions

We will compare PR to simple EM and to the methods of several previous studies in Section 6. Before that, we review the theory behind the previous work.

5.1 Bayesian Learning

The main learning method we will compare with experimentally is 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.

Recent advances in Bayesian inference methods have been applied to DMV grammar induction with varying levels of success. These approaches have focused on injecting linguistic knowledge 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 will review those methods; experimental comparisons are given in Section 6.

5.1.1 SPARSITY-INDUCING PRIORS

Dirichlet priors have been often used in DMV learning. More precisely, the prior distribution of the parameters of the DMV represented as a probabilistic context-free grammar (PCFG) is specified as a product of Dirichlets: $p(\theta) = \prod_{A \in V_N} D(\theta_A; \alpha_A)$ where the underlying CFG is $G = (V_N, V_T, R, S)$ with V_N, V_T, and R a set of non-terminals, terminals, and rules, respectively, and S a start symbol. (See Smith, 2006 for a detailed encoding of the DMV as a PCFG.) Each Dirichlet in this prior has the form:

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

where Z is a normalization term and the 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), this can be done efficiently with the product of Dirichlets type of prior. 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:

$$\textbf{EM M}\text{-step}: \theta_A^{t+1}(\beta) \propto \mathbf{E}_{q^{t+1}}[\#_{A \to \beta}(\mathbf{Y})],$$
$$\textbf{Dirichlet M}\text{-like-step}: \bar{\theta}_A^{t+1}(\beta) \propto \exp(\psi(\mathbf{E}_{q^{t+1}}[\#_{A \to \beta}(\mathbf{Y})] + \alpha_{A \to \beta})),$$

where ψ is the digamma function. As Figure 6 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 **Dirichlet M**-like-step, since small θ will get squashed to zero by the digamma.

This Dirichlet prior method is applied in several previous studies. Cohen et al. (2008) use this method for dependency parsing with the DMV and achieve improvements over basic EM. They set

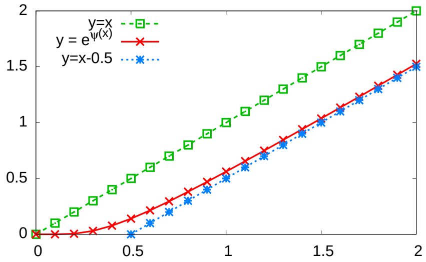


Figure 6: The digamma function.

Step	Learning Method	Formula
E-like	Standard EM	$q^{t+1} = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}\|\mathbf{X}))$
	Dirichlet Prior	Same as standard EM, but with $\overline{\theta}^t$ replacing θ^t
	PR	$q^{t+1} = \arg\min_q \mathbf{KL}(q(\mathbf{Y}) \parallel p_{\theta^t}(\mathbf{Y}\|\mathbf{X})) + \sigma \left\|\left\|\mathbf{E}_q[\phi(\mathbf{X},\mathbf{Y})]\right\|\right\|_\beta$
M-like	Standard EM	$\theta^{t+1} \propto \mathbf{E}_{q^{t+1}}[\log p_\theta(\mathbf{X},\mathbf{Y})]$
	Dirichlet Prior	$\overline{\theta}^{t+1} \propto \exp(\psi(\mathbf{E}_{q^{t+1}}[\log p_\theta(\mathbf{X},\mathbf{Y})] + \alpha))$
	PR	Same as standard EM

Table 1: E-like and M-like steps for the three main learning methods we compare in this work. The main differences are that PR changes the standard E-step to add a penalty term, while a Dirichlet prior changes the standard M-step to add pseudo-counts.

all hyperparameters to 0.25, resulting in a sparsifying prior (this is the method referred to as VB-Dirichlet in their work). In this paper we will refer to our own implementation of this method as the "sparsifying Dirichlet prior" (SDP) 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 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.

Headden III et al. (2009) also use a Dirichlet prior 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. We will refer to this (non-sparsifying) Dirichlet prior method as DP in the remainder of this paper. We have now covered the two learning methods we will directly compare to, EM and Dirichlet priors, so we summarize their respective E-like and M-like steps along with those of PR in Table 1 for ease of comparison.

5.1.2 PARAMETER-TYING PRIORS

In addition to Dirichlet priors, other types of priors have been used, namely logistic normal priors (LN) (Cohen et al., 2008) and shared logistic normal priors (SLN) (Cohen and Smith, 2009). While the SDP aims to induce parameter sparsity, LN and SLN aim to tie parameters together, but all of the methods have the same goal of favoring more concise grammars. By tying parameters for different tags, 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.

5.2 Other Learning Approaches

Several additional training alternatives have been proposed besides Bayesian methods. In particular, we will briefly describe here four such methods: contrastive estimation (CE), skewed deterministic annealing (SDA), structural annealing (SA), and direct model minimization through an integer program. We present an empirical comparison to the first three of these methods in Section 6 and show we can often achieve superior performance with posterior regularization. The fourth method has not yet been applied to the dependency parsing task we evaluate on in this work, so we defer direct comparison.

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

$$\log \prod_i \frac{\sum_{\mathbf{y} \in \mathbf{Y}} \exp(\theta \cdot f(\mathbf{x}^{(i)}, \mathbf{y}))}{\sum_{(\mathbf{x},\mathbf{y}) \in N(\mathbf{x}^{(i)}) \times Y} \exp(\theta \cdot f(\mathbf{x}, \mathbf{y}))}, \tag{9}$$

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 9) 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.

A final related learning approach is that of Ravi et al. (2010). This work attempts to directly minimize the number of tag bigrams for a supertagging task starting from the ending point of EM, then applying first one simple integer program, then a second more complex integer program. This method is similar to ours in that instead of using a prior, it attempts a direct minimization of tag pairs. One natural way to adapt it to dependency parsing would be to have an integer program that minimizes the number of parent-child tag pairs subject to the constraint that every sentence can still be assigned a complete parse tree. We do not compare to this proposed adaptation directly, but suspect that it would produce somewhat similar results to our PR method. One difference would be that while PR is very tightly integrated with EM, trading off between EM and the integer program would not be as straightforward as tuning a single hyperparameter.

5.3 Model Extensions

Before discussing experimental results, we detour to describe the extensions to the basic DMV that we experimented with. 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. With our experiments on these extended models, we aim to show that PR also achieves significant gains over other methods in a more complex model space.

5.3.1 EXTENDING STOP PROBABILITIES

The first 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 $0, 1, \ldots, S-2$, and $\geq S-1$ dependents in a given direction. The basic DMV has maximum stop valency 2 because it distinguishes between having zero dependents and at least one dependent in a given direction. A model with maximum stop valency of 3 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 $p_{stop}(\cdot \mid p, d, 2)$.

5.3.2 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 $0, 1, \ldots, C-2$, and $\geq C-1$ dependents in a particular direction. The basic DMV has maximum child valency 1 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 10).

With this model extension, the order in which dependents are generated becomes relevant to the probability of an overall parse tree. We choose to follow the standard inwards generation order. 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.

5.3.3 COMPLETE MODEL

Formally, under the extended DMV the probability of a sentence with POS tags **x** and dependency tree **y** is given by:

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

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 valency. 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:

$$\lambda p_{child}(y_c \mid y_p, y_d, y_{v_c}) + (1-\lambda) p_{child}(y_c \mid y_d, y_{v_c}), \tag{10}$$

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). 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 $\lambda = 1/3$.

6. Experiments

In this section we present positive experimental results validating the PR method. In Section 6.2 we detail experiments with different regularization strengths σ on English and analyze the correlation

	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	5K	24K	13K	2K	5K	0.4K	12K	7K	2K	3K	0.5K	3K
word types	11K	40K	20K	6K	10K	3K	2K	11K	7K	8K	3K	10K
word tokens	27K	139K	77K	11K	37K	2K	43K	43K	14K	23K	3K	18K

Table 2: Training 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.

between accuracy and the PR learning curves. The maximum accuracy we achieve is 64.5% using an E-DMV with PR-S and $\sigma = 160$. This is significantly above the best result of the SDP baseline, which is only 53.6%. In Section 6.3 we present a summary of related work, attempting to categorize the many dimensions along which researchers have explored modifications to the most basic EM DMV setup. While direct comparison of accuracy numbers from all related work is difficult, we present evidence that combining PR with a few of those modifications (for example random pool initialization) would result in the best accuracy yet achieved, especially for longer sentences. In Section 6.4 we apply PR to 11 additional languages, using English to select the regularization strength. Our multi-lingual results show that the PR method is indeed very broadly applicable. Averaging over all languages, there seem to only be minor differences in accuracy between PR-S and PR-AS, and both produce approximately equally sparse grammars. Under the DMV, PR-AS beats the SDP baseline for 10 out of 12 languages, Danish (Dk) and Swedish (Se) being the exceptions.

We conclude this overview of the experiments with two key points that we feel show PR to be a very useful and robust method for improving unsupervised dependency parsing:

- All except one of the 60 PR settings we try for English result in higher accuracy than the *best* SDP setting.

- In our multi-lingual experiments PR makes an average absolute accuracy gain of 5% over SDP for the DMV model.

6.1 Corpora

We evaluated 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 [Si] (Džeroski et al., 2006), and Turkish [Tr] (Oflazer et al., 2003). For English we trained on sections 2-21 of the Penn Treebank and tested on section 23. For the other languages, our training and test sets were exactly those used in CoNLL X shared task. Following Smith and Eisner (2006), we stripped punctuation from the sentences and kept only those sentences of length ≤ 10. Table 2 shows the size of the different training corpora after that filtering.

6.2 Results on English

We start with a comparison between EM and the two sparsity-inducing methods, PR and the sparsifying Dirichlet prior (SDP), on the English corpus. For all models we train for 100 iterations. Following Klein and Manning (2004), we use a "harmonic initializer", which we will refer on this paper as K&M. This initialization uses the posteriors of a "pseudo" E-step as initial parameters: 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.

At the end of training, 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 the attachment accuracy—the fraction of words assigned the correct parent—of their Viterbi (best) parses. We compare the performance of all training procedures both on the original DMV model as well as on the extended model E-DMV.

In Graça et al. (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. Thus, the following results do not use projection at decode time.

Following Cohen et al. (2008) we search for the best sparsifying parameter α for SDP training. See Table 5 in Appendix A for more details on the search for α. We find as Cohen et al. (2008) did that 0.25 is optimal for the DMV. SDP only achieves accuracy 46.4 in this setting, and even in its best E-DMV setting (V_s-V_c= 4-4, $\alpha = 0.1$), it only reaches accuracy 53.6. These values are far below most of the PR accuracies we will now discuss.

A comparison between EM and PR for both DMV and E-DMV are shown in Table 3. PR always performs better than EM. We performed a grid search over regularization strength (80 to 180 with a step of 20), for both the PR-S (symmetric constraint) and PR-AS (asymmetric constraint) formulations. A first observation based on Table 3 is that PR-S generally performs better than the PR-AS. Furthermore, PR-S seems less sensitive to the particular regularization strength. Comparing PR-S to EM, PR-S is always better, independent of the particular σ, with improvements ranging from 8% to 16%. The PR-AS constraints are also always better than EM for each model configuration and for all different parameter configurations. Note that the optimal parameter σ depends on the particular model configuration (V_s-V_c).

6.2.1 INSTABILITY WITH RESPECT TO σ

We can give a little more insight as to why we see some instability in the results with respect to the regularization strength. Figure 7 shows the accuracies on the English corpus broken down by POS tag category. 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 ultimately only achieve limited parsing accuracy.

Model	EM	PR						
	DMV							
		σ	80	100	120	140	160	180
2-1	45.8	PR-S	60.5	60.9	**62.0**	61.4	61.4	61.6
		PR-AS	53.8	54.3	55.3	54.3	54.6	54.6
V_s-V_c	E-DMV							
2-1	45.1	PR-S	60.7	59.9	61.3	61.6	62.1	60.2
		PR-AS	51.6	54.5	55.0	**62.4**	54.7	54.5
2-2	54.4	PR-S	**62.4**	57.1	57.8	57.6	57.1	58.8
		PR-AS	56.0	56.2	56.6	57.0	57.2	59.0
3-3	55.3	PR-S	59.3	60.8	60.0	62.2	**64.5**	64.1
		PR-AS	59.3	60.0	60.3	60.7	55.8	57.9
4-4	55.1	PR-S	59.4	61.2	61.6	63.9	**64.3**	63.6
		PR-AS	59.5	59.5	61.4	57.7	58.2	58.2

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_s-V_c used. Columns represent different σ for both constraints PR-S on the left and PR-AS on the right.

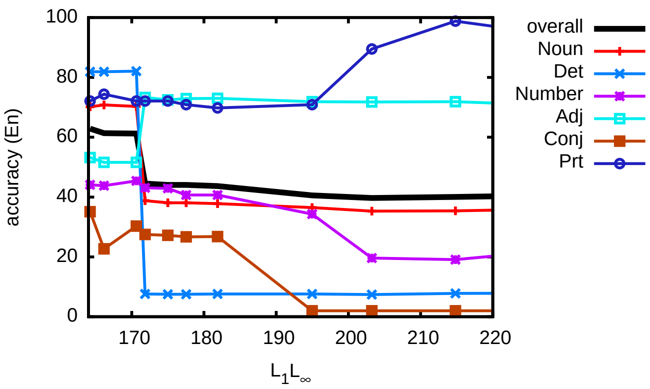


Figure 7: The accuracy overall and for different POS tag types in the English corpus as a function of ℓ_1/ℓ_∞ as we vary the constraint strength. EM has ℓ_1/ℓ_∞ of 431.17.

6.2.2 LEARNING CURVES

The top half of Figure 8 shows how accuracy and the various objective values change on a held-out development corpus for the DMV. (In all experiments, we held out the last 100 sentences of each training corpus for development; the numbers in Table 2 correspond to this reduced training set size. As we will discuss below they were unfortunately not reliable for picking hyperparameters.) First considering EM, we see that its accuracy is very stable after 20 iterations; its maximum value is at 80 iterations, but this is only marginally different from the value at 20 iterations. Its corresponding negative dev log likelihoood hits a minimum around 15 iterations, which correlates fairly well with accuracy, but then negative dev log likelihood steadily increases after this. So, while dev likelihood would select a reasonable stopping point in this case, it can hardly be said to generally correlate well with accuracy. Next, considering SDP, we see its accuracy is mostly stagnant after 25 iterations, yet its negative dev log likelihood continues to steadily decrease long past iteration 25. Thus, the value of the objective on the dev set for SDP does not provide a way to select a good stopping point, nor does it correlate particularly well with accuracy. Finally, considering PR, we see slightly noisier accuracy curves that take a little longer to reach their maximums: around iteration 30 for PR-S and iteration 40 for PR-AS. The PR dev objective value curves matches the behavior of the accuracy curve fairly well and would select a good iteration for stopping. In summary, for the DMV, dev likelihood would not be a bad proxy for selecting stopping points for EM and PR.

However, the correlation is not as good when using the extended models, whose learning curves are shown in the bottom half of Figure 8. For example, both PR-S and PR-AS experience large jumps in accuracy that are not reflected in the likelihood curves. Thus, in the remainder of this work we do not attempt to select a stopping point based on dev likelihood, but rather simply run all experiments for 100 iterations.

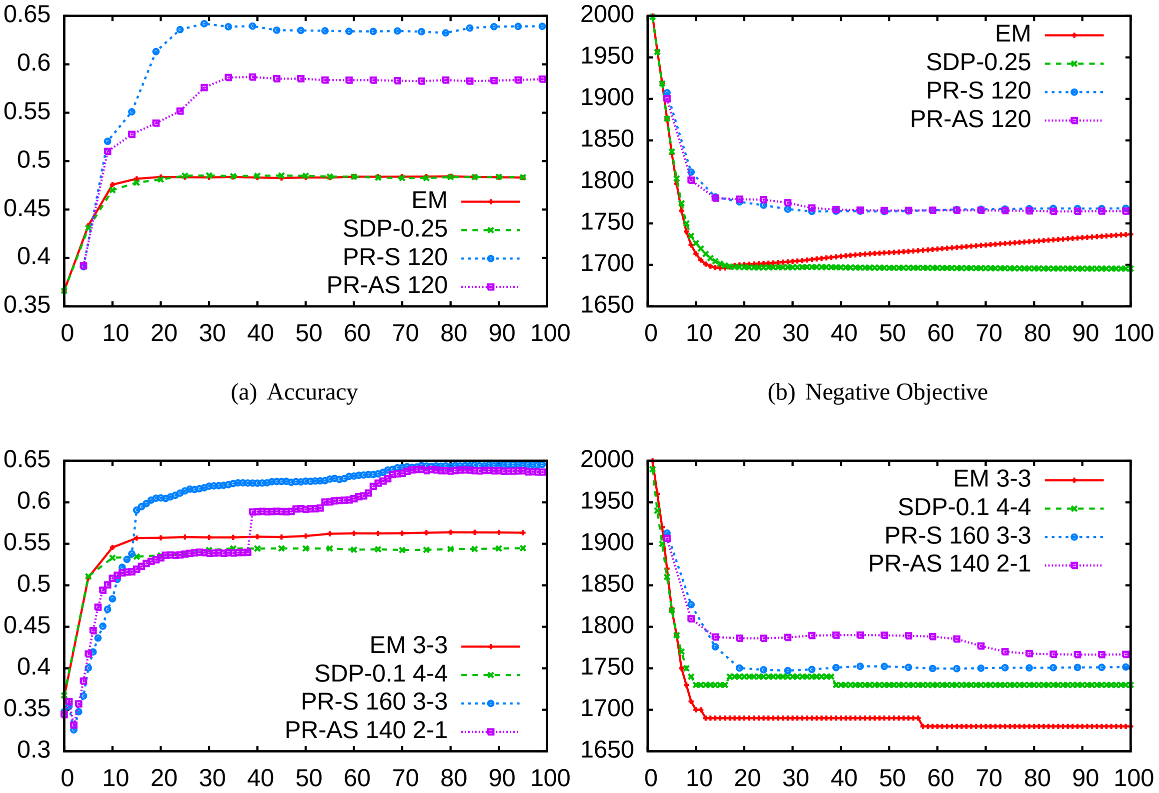
We also tried selecting a stopping point based on constituent contexts, motivated by Reichart and Rappoport (2009). Our hypothesis was that entropy of the distribution over contexts for each constituent should be small when parsing accuracy was high. However, comparing entropy of the gold trees to entropy of the trees produced by EM, this was only true for about half of the languages we tested on, and not strongly so for most of these. Also we note that we found no correlation between the PR objective on the development set and the best setting for the PR constraint strength, which does make it hard to pick this strength parameter in an unsupervised setting.

6.3 Comparison with Previous Work

Most results from previous work are not directly comparable due to differences in initialization, decoding method, or the incorporation of some degree of supervision. For this reason, we present the majority of the comparisons in Appendix B, where we also note implementation differences that we were able to determine. Here, we highlight the most salient accuracy numbers for the methods we mentioned in Section 5.

The best result reported thus far without additional lexical or multilingual information is that of Headden III et al. (2009). With a non-sparsifying Dirichlet prior and a learned (as opposed to constant) λ, they report an accuracy of 65.0 ($\pm$5.7)% for an an E-DMV of complexity $V_s = 2$, $V_c = 2$. (The $\pm$5.7 is a result of their use of a random pools initialization strategy.) We are able to achieve 64.5% accuracy with PR. We hypothesize that if PR were tested with random pools initialization and a learned λ, it would be able to make even further gains in accuracy. As noted in Appendix B, the learning of the smoothing parameter performed by Headden III et al. (2009) probably increases accuracy by about 5.5%. Similarly, Table 6 shows that random pools initialization tends to perform much better than the deterministic K&M initialization we use.

Other learning methods such as those discussed in Section 5 achieve slightly lower accuracies. We note that it is difficult however to make a complete comparison to them, as they operate only on the DMV model, not on any extended versions. Further, there are differences in the decoding method used. For example, the maximum accuracy achieved using shared logistic normal (SLN) priors with is 61.3% (Cohen and Smith, 2009). This is on the DMV model, where PR's maximum accuracy is a comparable 62%. But the SLN work uses MBR decoding and states its performance



(a) Accuracy

(b) Negative Objective

(c) Accuracy

(d) Negative Objective

Figure 8: Directed accuracy and the objective values on held-out development data as a function of the training iteration for the DMV (top) and E-DMV (bottom) with the best parameter settings.

is better than that of the Viterbi that we use. So, comparisons should be taken with a grain of salt. Comparing to contrastive estimation and annealing methods, accuracies are further below those of PR. With the DMV model and K&M initialization: CE is 48.7%, SDA is 46.7%, and SA is 51.5%. For a more extensive comparison to experimental results from related work, see Appendix B.

6.4 Results on Other Languages

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

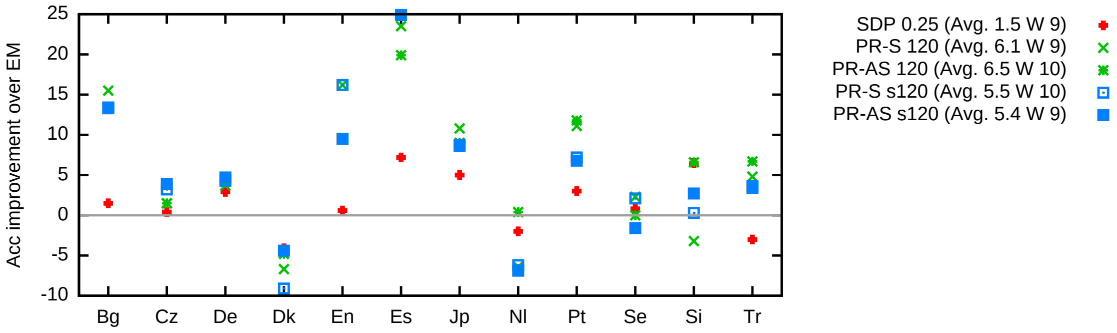


Figure 9: Difference in accuracy between the sparsity inducing training methods and EM training for the DMV model across the 12 languages. Avg: Average improvement over EM. W: Number of languages better than EM.

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.

For a table of exact accuracy numbers, we refer the reader to Table 7 in Appendix C. In this section we provide some figures illustrating the most salient aspects of the results from this table. Figure 9 illustrates the differences between the EM training and the different sparsity inducing training methods for the DMV. The zero line in Figure 9 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 9. Figure 10 shows a similar comparison of the PR methods with respect to a SDP learning baseline. We see in Figure 10 that PR is better than SDP for most languages. Figure 11 compares the differences of each training method against EM training using the E-DMV model with the best setting found for English. Both PR-S and PR-AS perform better than EM in most cases. The average improvement is even bigger for PR-S than under the DMV, but PR-AS does not make such large gains. This is probably due to the selection of a simpler model for PR-AS (V_s-V_c= 2-1). While this simpler model performed better than the more complex ones for English, this does not generalize to all languages.

Figure 12 compares the different sparsity approaches. On the left we compare PR-S versus PR-AS without scaling on the DMV. PR-AS beats PR-S in 6 out of 12 cases and the two methods tie in one case (Czech). Over all 12 languages, the average difference between PR-AS and PR-S is only 3.2% on the DMV. We note that the difference is bigger for the E-DMV models, but this is possibly due to the selection of a simpler model (V_s-V_c = 2-1) for PR-AS. On the right side of the same figure, we compare PR-AS without scaling versus PR-AS with scaling. The unscaled

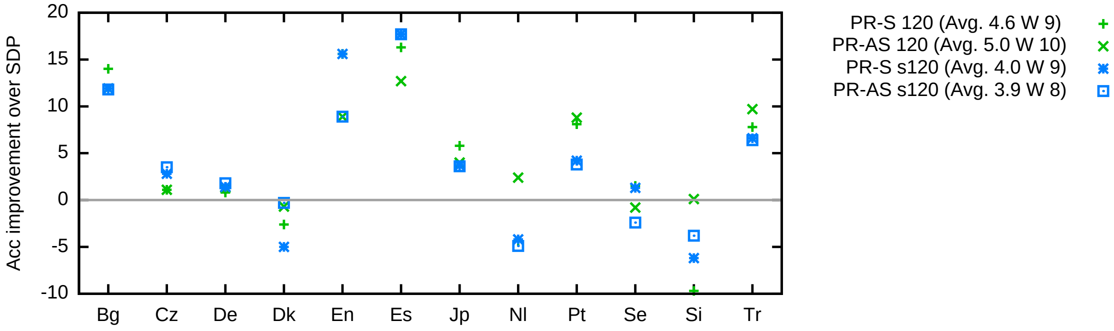


Figure 10: Difference in accuracy between PR training with the different constraints and SDP for the DMV model across the 12 languages. Avg: Average improvement over SDP. W: Number of languages better than SDP.

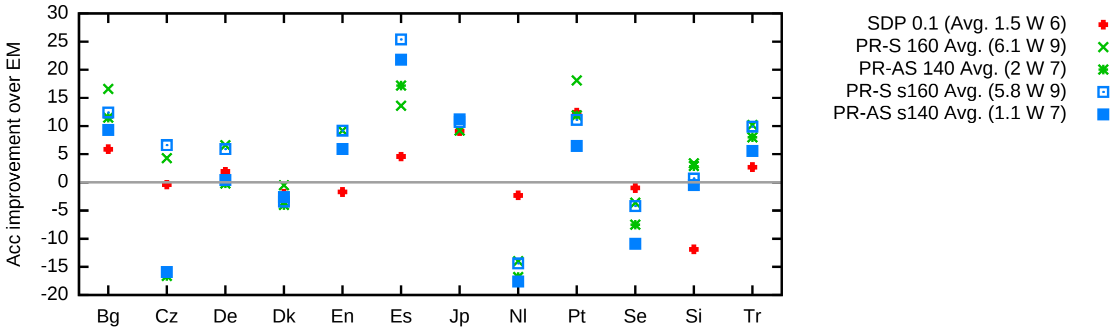
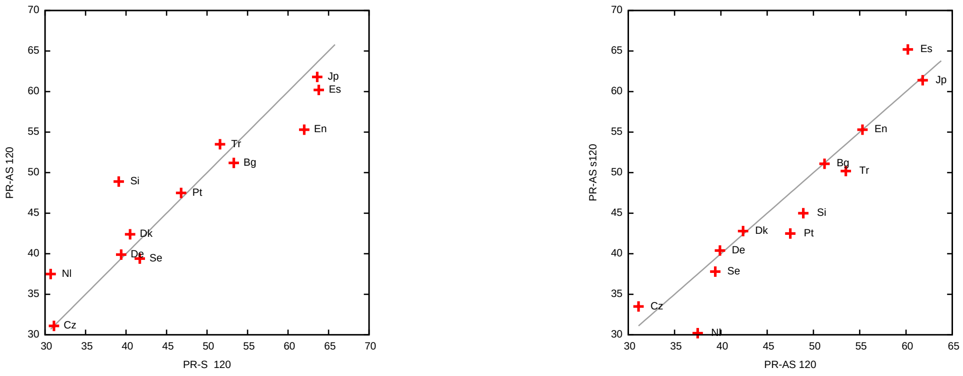


Figure 11: Difference in accuracy between the sparsity inducing training methods and EM training for the E-DMV model with the different training method across the 12 languages. Avg: Average improvement over EM. W: Number of languages better than EM.

version tends to perform better. In general, scaling that increases the constraint strength seems to be advantageous, the exception being for Dutch (Nl). Increased strength tends to correlate with increased runtime though, so there is a tradeoff to be made there.

Figure 13 compares the sparsity achieved by EM, SDP, and the PR methods on the DMV. We can see that the PR methods indeed achieve much greater sparsity than EM, and that SDP is only slightly more sparse than EM. If we also compared to supervised model initialization, most of the PR instances would have greater sparsity than the supervised, and EM and SDP would be much less sparse than the supervised. So, it seems that over-sparsifying is allowing us to achieve better accuracy than under-sparsifying. Although also not shown in the plot, we observe similar sparsity patterns on the test data as well.



(a) PR-S versus PR-AS

(b) No scaling versus scaling

Figure 12: Comparing the different sparsity constraints for the DMV model over twelve different languages. Left: PR-S vs PR-AS. Right: PR-AS without scaling vs PR-AS with scaling.

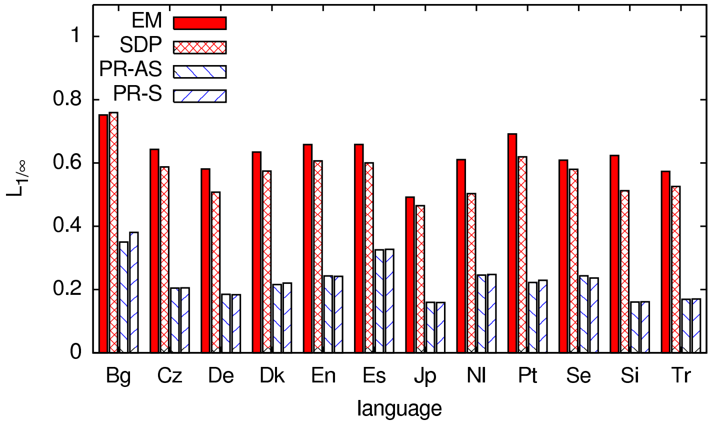


Figure 13: Comparing DMV grammar ambiguities on the training data by computing the average number of parent tags per child tag (ℓ_1/ℓ_∞ divided by number of child tags) and normalizing it by the theoretical maximum for each language. Grammar ambiguities from left to right within each group of bars are those resulting from: EM, SDP with $\alpha = 0.25$, PR-S with $\sigma = 120$, and PR-AS with $\sigma = 120$. Higher values imply less sparsity.

7. Analysis

Our accuracy numbers validate that PR is useful. In this section we attempt to analyze how and why it is useful, to validate our original claim that sparsity in parent-child types is the phenomenon we are capturing.

One common EM error that PR fixes in many languages is the directionality of the noun-determiner relation. Figure 14 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 er-

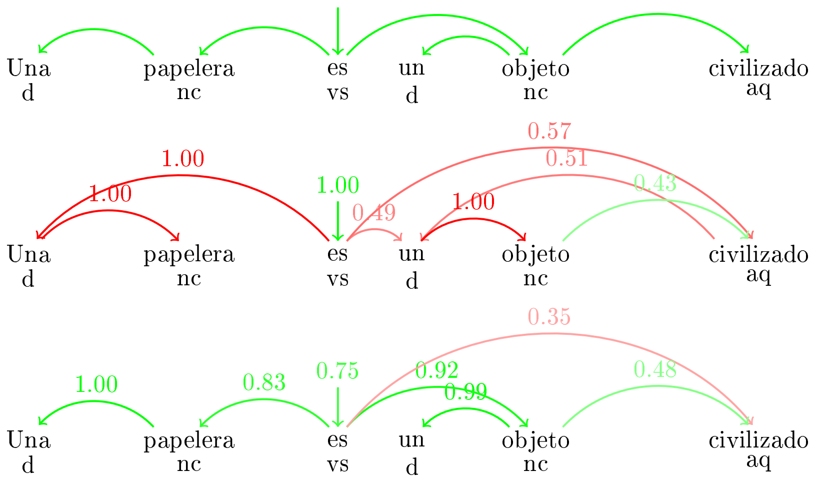


Figure 14: Posterior edge probabilities for an example sentence from the Spanish test corpus. **Top** is Gold, **middle** is EM, and **bottom** is PR.

ror. 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 (them)”) 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 does not come with a determiner.

Table 4 contrasts the most frequent types of errors EM, SDP, 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:

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

In the following subsections we provide some analysis of the results from Table 4.

7.1 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 SDP 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

	EM			SDP			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 4: 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.

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 SDP.

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 SDP have 0 accuracy for this key.

There are a couple of errors common to EM, SDP, 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.

7.2 Bulgarian Corrections

Moving beyond English, we 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 SDP 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 SDP 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 same reasoning about PR that we used before, 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 SDP do.

Although PR is able to make great improvements for Bulgarian parsing, it is clearly crippled by the small number of POS tags. EM, SDP, 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 4 to the reader.

8. 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 sparsifying 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 sparsifying 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 improves over EM by an average of 6.5% and beats EM by at least 1% on 9 out of 12 languages. It also improves over the Bayesian learning approach by an average of 5% with gains of more than 1% for for 9 out of 12 languages.

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 PR objective on held-out development sets does not seem to be a reliable proxy for the model quality. Similarly, additional unsupervised measures for parse quality, motivated by the work of Reichart and Rappoport (2009) on counting constituent contexts, were unreliable. Even in the absence of a good unsupervised measure of model quality, a better method 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. It would be worth initializing our models with the techniques proposed by Spitkovsky et al. (2010), since they produce better results, are deterministic, and reduce the number of parameters that need to be tuned. Following the spirit of those approaches approaches, we also suggest 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.

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful comments. J. Gillenwater was partially supported by NSF-IGERT 0504487. K. Ganchev was supported by ARO MURI SUBTLE W911NF-07-1-0216. J. V. Graça was supported by a fellowship from Fundação para a Ciência e Tecnologia (SFRH/ BD/ 27528/ 2006) and by FCT project CMU-PT/HuMach/0039/2008. B. Taskar was partially supported by DARPA CSSG and ONR Young Investigator Award N000141010746.

Appendix A. Choosing the SDP Hyperparameter

We tried four different values for α: $\{0.01, 0.1, 0.25, 1\}$. (Note that the value 1 actually results in a non-sparsifying prior; this setting is not as good as the sparsifying, as Table 5 shows.)

Table 5 shows the directed accuracy for both the DMV and the E-DMV models trained using EM and SDP. We see in Table 5 that the extended model generally outperforms the DMV, for both EM and SDP. However, we also see that SDP does not always help: for all valences tried for the E-DMV except $(V_s, V_c) = (2, 1)$, 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 held-out development set converged. Comparing the performance of the training methods, we see that for the DMV model, SDP 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 SDP is slightly lower than the one reported in that paper, probably due to different stopping criteria during training.

Appendix B. Extended Comparison to Related Results

In this appendix we present a more extensive comparison between the performances of different models described in the literature for unsupervised dependency parsing. Table 6 presents the accuracy values reported in various previous papers and the values for approaches tried in this paper.

		SDP $\alpha =$			
	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 5: Directed attachment accuracy results on the test corpus (for sentences of lengths ≤ 10, no punctuation). The second column gives EM results, and the other columns are SDP 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_s-V_c). Note that the 2-1 model is just the DMV plus smoothing of the child probabilities with $\lambda = 0.33$. Bold represents the best parameter setting both for the DMV model and the E-DMV model.

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. The point of this section is mostly to highlight the many different variations of the DMV training and modeling that have been tried in the past. Table 6 is meant as a resource for comparing some of the best accuracies that these methods have achieved. It is hard to draw any sweeping conclusions from these numbers, but we hope that this summary of related work helps future work by suggesting reasonable choices for initialization, model complexity, smoothing, and other modeling decisions.

We start by comparing the effects of different initialization procedures. (See entries 1-6 in Table 6.) Although orthogonal to the learning procedure used, these differences are significant to keep in mind when comparing to previous work. We compare the results on the DMV. First we compare to work by Headden III et al. (2009) using random pools initialization. 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.

The other initializations compared in Table 6 are from recent work by Spitkovsky et al. (2010). These initialization methods aim to gradually increase the complexity of a model, as measured by the size of the search space, which for the DMV model is exponential in sentence length. The Baby Steps (BS) method starts by training the model on sentences of length 1, then the parameters of this model are used to initialize a training run over sentences of length 2, and so on. 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. A hybrid method Leapfrog (LP) combines the models from the two previous approaches. All of these methods also seem to improve over the K&M initialization.

We note that there are some differences in the setup of the various initialization experiments: the model initialized with RandomP described in Headden III et al. (2009) is trained using a Dirichlet prior with a hyperparameter of 1 (non-sparsifying DP), while all the other models are trained using EM. Additionally, the models from Spitkovsky et al. (2010) use a larger amount of data. Nonetheless, it seems likely that if we combined some of these initializations with out PR method, we would see even better performance than with the K&M setup that we use for simplicity in our current experiments.

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 (compare entry 2 to entry 7 and entry 1 to entry 8).

Entries 9 to 20 compare different training approaches for the basic DMV. Entry 9 corresponds to training the model with SDP 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 do not 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 selection of training parameters (several values were tried, and the one that produced the highest accuracy on the test data was selected). 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 27, 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, SDP, and PR with both types of constraints. Model selection in these cases is supervised, based on accuracy for the ≤ 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 SDP on sentences of lengths ≤ 10, and an even bigger improvement for longer sentences. In entries 25 to 27 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 difference 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 6 doesn't report results for the papers that use extra information. Namely, 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

	Init	Training	Model	Directed			Undirected		
				≤ 10	≤ 20	all	≤ 10	≤ 20	all
Model Initialization									
1	K&M	EM	DMV	45.8	40.2	35.9	63.4	58.0	54.2
2	RandomP	DP	DMV	55.7 (±8.0)					
3	BS	Ad-Hoc @15	DMV	55.5	44.3	39.2			
4	BS	Ad-Hoc @45	DMV	55.1	44.4	39.4			
5	LsM	Ad-Hoc @15	DMV	56.2	48.2	44.1			
6	LP	Hybrid @45	DMV	**57.1**	**48.7**	**45.0**			
Smoothing effects									
7	RandomP	DP	DMV (λ learned)	**61.2 (**±1.2**)**					
8	K&M	EM	DMV (λ = 0.33)	45.1	38.7	34.0	62.7	56.9	52.7
DMV									
9	K&M	SDP 0.25 *	DMV	46.4	40.9	36.5	64.0	58.6	54.8
10	K&M	PR-S 120 *	DMV	**62.0**	**53.8**	**48.9**	**69.8**	**62.4**	**58.2**
11	K&M	PR-AS 120 *	DMV	55.3	49.4	44.4	67.1	60.7	56.4
12	K&M	LN I	DMV	56.6	43.3	37.4			
13	K&M	LN families	DMV	59.3	45.1	39.0			
14	K&M	SLN Tie V	DMV	60.2	46.2	40.0			
15	K&M	SLN Tie N	DMV	60.2	46.7	40.9			
16	K&M	SLN Tie V & N	DMV	61.3	47.4	41.4			
17	K&M	SLN Tie A	DMV	59.9	45.8	40.9			
18	K&M	CE *	DMV	48.7			64.9		
19	K&M	SDA *	DMV	46.7			64.3		
20	K&M	SA *	DMV	51.5			67.9		
E-DMV									
21	K&M	EM	E-DMV(3,3) (λ = 0.33) *	55.3	46.4	42.6	69.0	61.9	58.3
22	K&M	SDP 0.1 *	E-DMV(4,4) (λ = 0.33) *	53.6	43.8	39.6	67.5	59.0	54.9
23	K&M	PR-S 160 *	E-DMV(3,3) (λ = 0.33) *	64.5	**54.6**	**49.5**	69.9	60.9	56.0
24	K&M	PR-AS 140 *	E-DMV(2,1) (λ = 0.33) *	62.2	53.2	48.5	**70.8**	**61.9**	**57.8**
25	K&M	EM	E-DMV(2,2)	56.5			69.7		
26	RandomP	DP	E-DMV(2,2)	53.3 (±7.1)					
27	RandomP	DP	E-DMV(2,2) (λ learned)	**65.0 (**±5.7**)**					

Table 6: Comparison with previous published results. Results for entries 3, 4, 5, and 6 are taken from Spitkovsky et al. (2010), entries 2, 7, 26, and 27 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). A star (*) in the training column indicates supervised selection of training parameters (PR regularization strength, SDP prior hyperparameter, etc.); a star in the model column indicates supervised selection of model complexity.

	Bg	Cz	De	Dk	En	Es	Jp	Nl	Pt	Se	Si	Tr	Avg
	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	40.9
SDP 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	42.4
PR-S 120	**53.3**	31.1	39.4	40.5	**62.0**	63.8	**63.6**	30.7	46.8	**41.7**	39.1	51.6	47.0
PR-AS 120	51.2	31.1	39.9	42.4	55.3	60.2	61.8	**37.5**	**47.5**	39.4	**48.9**	**53.5**	47.4
PR-S s120	51.2	32.8	40.0	38.1	**62.0**	**65.2**	61.5	30.9	42.9	41.5	42.6	50.4	46.6
PR-AS s120	51.1	**33.5**	**40.4**	42.8	55.3	**65.2**	61.4	30.2	42.5	37.8	45.0	50.2	46.3
	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	45.9
SDP-(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	47.4
PR-S-(3,3) 160	**58.3**	53.2	**46.7**	45.9	**64.5**	57.9	57.7	33.5	**54.0**	45.0	**50.9**	**56.4**	52.0
PR-AS-(2,1) 140	53.2	32.3	39.9	42.4	61.2	61.5	59.6	30.7	47.8	41.1	50.4	54.2	47.9
PR-S-(3,3) s160	54.1	**55.5**	46.0	43.0	**64.5**	**69.7**	59.2	33.1	47.0	44.4	48.2	56.1	51.7
PR-AS-(2,1) s140	51.0	33.0	40.5	43.8	61.2	66.1	**59.7**	29.9	42.4	37.7	47.0	51.8	47.0
	Scaled Strengths												
English $\sigma = 120$	88	451	249	35	120	8	140	138	47	75	10	57	118
English $\sigma = 140$	103	526	290	41	140	9	163	161	55	88	11	67	138
English $\sigma = 160$	118	602	332	47	160	11	187	185	62	100	13	76	158

Table 7: 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. **SDP**: Sparsifying 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.

≤ 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. Thus, we think that PR is a very promising technique for use with other data sets, where longer sentences are common.

Appendix C. Multilingual Results in Table Form

Table 7 shows the performance for all models and training procedures for the 12 different languages.

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*, 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*, 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. TLT*, 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*, 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.

S. Ravi, J. Baldridge, and K. Knight. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proc. ACL*, 2010.

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. 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.