A convex and feature-rich discriminative approach to dependency grammar induction

Édouard Grave
Columbia University
edouard.grave@gmail.com

Noémie Elhadad
Columbia University
noemie.elhadad@columbia.edu

Abstract

In this paper, we introduce a new method for the problem of unsupervised dependency parsing. Most current approaches are based on generative models. Learning the parameters of such models relies on solving a non-convex optimization problem, thus making them sensitive to initialization. We propose a new convex formulation to the task of dependency grammar induction. Our approach is discriminative, allowing the use of different kinds of features. We describe an efficient optimization algorithm to learn the parameters of our model, based on the Frank-Wolfe algorithm. Our method can easily be generalized to other unsupervised learning problems. We evaluate our approach on ten languages belonging to four different families, showing that our method is competitive with other state-of-the-art methods.

1 Introduction

Grammar induction is an important problem in computational linguistics. Despite having recently received a lot of attention, it is still considered to be an unsolved problem. In this work, we are interested in unsupervised dependency parsing. More precisely, our goal is to induce directed dependency trees, which capture binary syntactic relations between the words of a sentence. Since our method is unsupervised, it does not have access to such syntactic structure and only take as input a corpus of words and their associated parts of speech.

Most recent approaches to unsupervised dependency parsing are based on probabilistic generative models, such as the dependency model with valence introduced by Klein and Manning (2004). Learning the parameters of such models is often done by maximizing the log-likelihood of unlabeled data, leading to a non-convex optimization problem. Thus, the performance of those methods rely heavily on the initialization, and practitioners have to find good heuristics to initialize their models.

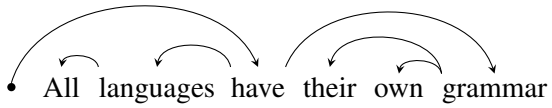

Figure 1: An example of dependency tree.

In this paper, we describe a different approach to the problem of dependency grammar induction, inspired by discriminative clustering. We propose to use a feature-rich discriminative parser, and to learn the parameters of this parser using a convex quadratic objective function. In particular, this approach also allows us to induce non-projective dependency structures. Following the work of Naseem et al. (2010), we use language-independent rules between pairs of parts-of-speech to guide our parser. More precisely, we make the following contributions:

- Our method is based on a feature-rich discriminative parser (section 3);
- Learning the parameters of our parser is achieved using a convex objective, and is thus not sensitive to initialization (section 4);
- Our method can produce non-projective dependency structures (section 3.2.2);
- We propose an efficient algorithm to optimize the objective, based on the Frank-Wolfe method (section 5);
- We evaluate our approach on the universal treebanks dataset, showing that it is competitive with the state-of-the-art (section 6).

2 Related work

A lot of research has been carried out in the last decade on dependency grammar induction. We review the dependency model with valence, on which most unsupervised dependency parsers are based, before presenting different extensions and learning algorithms. Finally, we review discriminative clustering, on which our method is based.

DMV. The dependency model with valence (DMV), introduced by Klein and Manning (2004), was the first method to outperform the baseline consisting in attaching each token to the next one. The DMV is a generative probabilistic model of the dependency tree and parts-of-speech of a sentence. It generates the root first, and then recursively generates the tokens down the tree. The probability of generating a new dependent for a given token depends on the direction (left or right) and whether a dependent was already generated in that direction. Then, the part-of-speech of the new dependent is generated according to a multinomial distribution conditioned on the direction and the head's POS.

Extensions. Several extensions of the dependency model with valence have been proposed. Headden III et al. (2009) proposed the lexicalized extended valence grammar (EVG), in which the probability of generating a POS also depends on the valence information. They rely on smoothing to tackle the increased number of parameters. Mareček and Žabokrtskỳ (2012) described an approach using a n-gram reducibility measure, which capture which words can be deleted from a sentence without making it syntactically incorrect. Cohen and Smith (2009) introduced a prior, based on the shared logistic normal distribution. This prior allowed to tie the grammar parameters corresponding to different POS belonging to the same coarse groups, such as all the POS corresponding to verbs. Berg-Kirkpatrick and Klein (2010) proposed to tie the parameters of grammars for different languages using a prior based on a phylogenetic tree. Naseem et al. (2010) proposed a set of rules between parts-of-speech, encoding *syntactic universals*, such as the fact that adjectives are often dependents of nouns. They used posterior regularization (Ganchev et al., 2010) to impose that a certain amount of the infered dependencies verifies one of these rules. Also using posterior regularization, Gillenwater et al. (2011) imposed a sparsity bias on the infered dependencies, enforcing a small number of unique dependency types. Finally, Blunsom and Cohn (2010) reformulated dependency grammar induction using tree substitution grammars, while Bisk and Hockenmaier (2013) proposed to use combinatory categorial grammars.

Learning. Different algorithms have been proposed to improve the learning of the parameters of the dependency model with valence. Smith and Eisner (2005) proposed to use constrastive estimation to learn the parameters of a log-linear parametrization of the DMV, while Spitkovsky et al. (2010b) showed that using Viterbi EM instead of classic EM leads to higher accuracy. Observing that learning from shorter sentences is easier (because less ambiguous), Spitkovsky et al. (2010a) presented different techniques to learn grammar from increasingly longer sentences. Gimpel and Smith (2012) introduced a model inspired by the IBM1 translation model for grammar induction, resulting in a concave log-likelihood function. They show that initializing the DMV with the output of their model leads to improved dependency accuracies. Hsu et al. (2012) and Parikh et al. (2014) introduced spectral methods for unsupervised dependency and constituency parsing. Finally, Spitkovsky et al. (2013) introduced different heuristics for avoiding local minima while Gormley and Eisner (2013) proposed a method to find the global optimum of non-convex problems, based on branch-and-bound.

Discriminative clustering. Our unsupervised parser is inspired by discriminative clustering, introduced by Xu et al. (2004). Given a set of points, the objective of discriminative clustering is to assign labels to these points that can be easily predicted using a discriminative classifier. Xu et al. (2004) introduced a formulation using the hinge loss, Bach and Harchaoui (2007) proposed to use the squared loss instead, while Joulin et al. (2010) proposed a formulation based on the logistic loss. Recently, a formulation based on discriminative clustering was proposed for the problem of distant supervision for relation extraction (Grave, 2014) and for the problem of finding the names of characters in TV series based on the corresponding scripts (Ramanathan et al., 2014). Closest to our approach, extensions of discriminative clustering were used to align sequences of labels or text with

videos (Bojanowski et al., 2014; Bojanowski et al., 2015) or to co-localize objects in videos (Joulin et al., 2014).

3 Model

In this section, we describe the parsing model used in our approach and briefly review the corresponding decoding algorithms. Following McDonald et al. (2005b), we propose to cast the problem of dependency parsing as a maximum weight spanning tree problem in directed graphs.

3.1 Edge-based factorization

Let us start by setting up some notations. An input sentence of length n is represented by an $n-$uplet $\mathbf{x} = (x_1, ..., x_n)$. The dependency tree corresponding to that sentence is represented by a $n \times (n+1)$ binary matrix $\mathbf{y}$, such that $y_{ij} = 1$ if and only if the head of the token i is the token j (and thus, the integer $n+1$ represents the root of the tree).

In this paper, we follow a common approach by factoring the score of dependency tree as the sum of the scores of the edges forming that tree. We assume that each pair of tokens (i, j) is represented by a high-dimensional feature vector $f(\mathbf{x}, i, j) \in \mathbb{R}^d$. Then, the score s_{ij} of the edge (i, j) is obtained using the linear model

$$s_{ij} = \mathbf{w}^\top f(\mathbf{x}, i, j),$$

where $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector. Thus the score s corresponding to the tree $\mathbf{y}$ is equal to

$$\begin{aligned} s &= \sum_{(i,j) \text{ s.t. } y_{ij}=1} s_{ij} \\ &= \sum_{(i,j) \text{ s.t. } y_{ij}=1} \mathbf{w}^\top f(\mathbf{x}, i, j). \end{aligned}$$

Assuming that the parameter vector $\mathbf{w}$ is known, parsing a sentence reduces to finding the tree with the highest score, which is the maximum weight spanning tree.

3.2 Maximum spanning trees

Different sets of spanning trees have been considered in the setting of supervised dependency parsing. We briefly review those sets, and describe the corresponding algorithms to compute the maximum weight spanning tree over those sets.

3.2.1 Projective dependency trees

First, we consider the set of projective spanning trees. A dependency tree is said to be projective if the dependencies do not cross when drawn above the words in linear order. Similarly, this means that word and all its descendants form a contiguous substring of the sentence. Projective dependency trees are thus strongly related to context free grammars, and it is possible to obtain the maximum weight spanning projective tree using a modified version of the CKY algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967). The complexity of this algorithm is $O(n^5)$. This led Eisner (1996) to propose an algorithm for projective parsing which has a complexity of $O(n^3)$. Similarly to CKY, the Eisner algorithm is based on dynamic programming, parsing a sentence in a bottom-up fashion. Finally, it should be noted that the dependency model with valence, on which most approaches to dependency grammar induction are based, produces projective dependency trees.

3.2.2 Non-projective dependency trees

Second, we consider the set of non-projective spanning trees. Indeed, many languages, such as Czech or Dutch, have a significant number of non-projective edges. In the context of supervised dependency parsing, McDonald et al. (2005b) shown that using non-projective trees improves the accuracy of dependency parsers for those languages. The maximum weight spanning tree in a directed graph can be computed using the Chu-Liu/Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which has a complexity of $O(n^3)$. Later, Tarjan (1977) proposed an improved version of this algorithm for dense graphs, whose complexity is $O(n^2)$, the same as for undirected graphs using Prim's algorithm. Thus a second advantage of using non-projective dependency trees is the fact that it leads to more efficient parsers.

4 Learning the parameter vector

In this section, we describe the loss function we use to learn the parameter vector $\mathbf{w}$ from unlabeled sentences.

4.1 Problem formulation

From now on, $\mathbf{y}$ is a vector representing the dependency trees corresponding to the whole corpus. Thus, each index i corresponds to a potential dependency between two words of a given sentence.

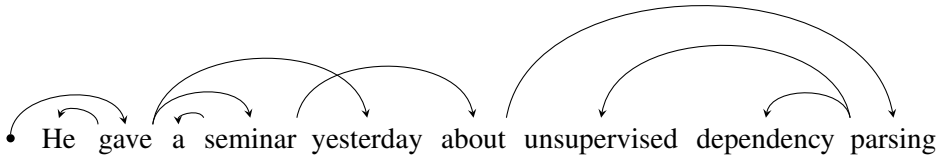

Figure 2: Example of a non-projective dependency tree in english.

Like before, $y_i = 1$ if and only if there is a dependency between those two words, and $y_i = 0$ otherwise. The set of dependencies that form valid trees is denoted by the set $\mathcal{T}$.

Inspired by the discriminative clustering framework introduced by Xu et al. (2004), our goal is to jointly find the dependencies represented by the vector $\mathbf{y}$ and the parameter vector $\mathbf{w}$ which minimize the regularized empirical risk

$$\min_{\mathbf{y}\in\mathcal{T}} \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \mathbf{w}^\top \mathbf{x}_i) + \lambda\Omega(\mathbf{w}), \quad (1)$$

where ℓ is a loss function and Ω is a regularizer. The intuition is that we want to find the dependency trees $\mathbf{y}$ that can be easily predicted by a discriminative parser, whose parameters are $\mathbf{w}$.

Following Bach and Harchaoui (2007), we propose to use the squared loss ℓ defined by

$$\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

and to use the ℓ_2-norm as a regularizer. In that case, we obtain the objective function:

$$\min_{\mathbf{y}\in\mathcal{T}} \min_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2. \quad (2)$$

One of the main advantages of using the squared loss is the fact that the corresponding objective function is jointly convex in $\mathbf{y}$ and $\mathbf{w}$. Indeed, the objective is the composition of an affine mapping, defined by $(\mathbf{y}, \mathbf{w}) \mapsto \mathbf{y} - \mathbf{X}\mathbf{w}$, with a convex function, defined by $\mathbf{u} \mapsto \mathbf{u}^\top\mathbf{u}$. Thus, the objective function is convex (see section 3.2.2 of Boyd and Vandenberghe (2004)). The problem (2) is thus non-convex only because of the combinatorial constraints on the binary vector $\mathbf{y}$, namely that $\mathbf{y}$ should represents valid trees.

4.2 Convex relaxation

The set $\mathcal{T}$ of vectors representing valid dependency trees is a finite set of binary vectors. We can thus take the convex hull of those points and denote it by $\mathcal{Y}$:

$$\mathcal{Y} = \text{conv}(\mathcal{T}).$$

VERB $\mapsto$ VERB	NOUN $\mapsto$ NOUN
VERB $\mapsto$ NOUN	NOUN $\mapsto$ ADJ
VERB $\mapsto$ PRON	NOUN $\mapsto$ DET
VERB $\mapsto$ ADV	NOUN $\mapsto$ NUM
VERB $\mapsto$ ADP	NOUN $\mapsto$ CONJ
ADJ $\mapsto$ ADV	ADP $\mapsto$ NOUN

Table 1: Set of universal rules used in our parser.

By definition, this set is a convex polytope. We then propose to replace the combinatorial constraints on the vector $\mathbf{y}$ by the fact that $\mathbf{y}$ should be in the convex polytope $\mathcal{Y}$. We thus obtain a convex quadratic program, with linear constraints, as follows:

$$\min_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2. \quad (3)$$

We will describe how to compute the optimal solution of this problem in section 5.

4.3 Rounding

Given a continuous solution $\mathbf{y}_c \in \mathcal{Y}$ of the relaxed problem, it is possible to obtain a solution of the integer problem by finding the tree $\mathbf{y}_d \in \mathcal{T}$ which is closest to $\mathbf{y}_c$, by solving the problem

$$\min_{\mathbf{y}_d\in\mathcal{T}} \|\mathbf{y}_d - \mathbf{y}_c\|_2^2.$$

The solution of the previous problem can easily be formulated is a minimum weight spanning tree problem. Indeed, by developping the previous expression, and using the fact that for all trees $\mathbf{y}_d \in \mathcal{T}$, $\mathbf{y}_d^\top\mathbf{y}_d = n$, where n is the number of tokens, the previous problem is equivalent to:

$$\min_{\mathbf{y}_d\in\mathcal{T}} -\mathbf{y}_d^\top\mathbf{y}_c,$$

whose solution is obtained using the minimum weight spanning tree algorithm. It should be noted that the rounding solution is not necessarily the optimal solution of the integer problem.

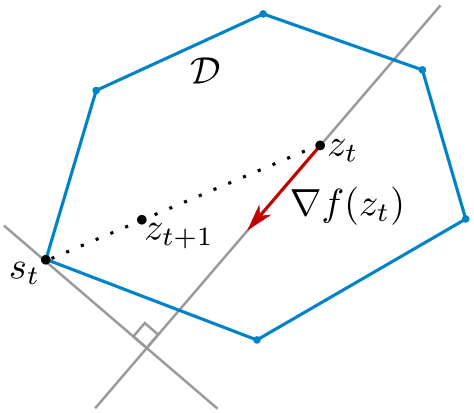

Figure 3: Illustration of a Frank-Wolfe step.

4.4 Prior on y

We now describe how to guide our unsupervised parser, by using universal rules. Following Naseem et al. (2010), we want a certain percentage of the infered dependencies to satisfy one of the twelve universal syntactic rules, listed in Table 1. Let $\mathcal{S}$ be the set of indices corresponding to word pairs that satisfy one of these rules. Then, imposing that a certain percentage c of dependencies satisfy one of those rules can be obtained by imposing the constraint:

$$\frac{1}{n}\sum_{i\in\mathcal{S}} y_i \geq c.$$

This linear constraint is equivalent to $\mathbf{u}^\top \mathbf{y} \geq c$, where the vector $\mathbf{u}$ is defined by

$$u_i = \begin{cases} 1/n & \text{if } i \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases}$$

Using Lagrangian duality, we can obtain the following equivalent penalized problem:

$$\min_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \mu\, \mathbf{u}^\top \mathbf{y}. \quad (4)$$

The penalized and constrained problems are equivalent, since for every c, there exists a μ such that the two problems have the same optimum. From an optimization point of view, it is easier to deal with the penalized problem and we will thus use it in the next section.

5 Optimization

One could use a general purpose quadratic solver to compute the solution of the previous convex problem. However, this might be inefficient since it does not use the structure of the polytope and, in particular, the fact that one can easily minimize a linear function over the tree polytope using the minimum weight spanning tree algorithm. Instead we propose to use the Frank-Wolfe algorithm, that we now describe.

Algorithm 1: Frank-Wolfe algorithm

for $t \in \{1, ..., T\}$ **do**
- Compute the gradient: $g_t = \nabla f(z_t)$
- Solve the linear program: $s_t = \min_{s\in\mathcal{D}} s^\top g_t$
- Take the Frank-Wolfe step: $z_{t+1} = \gamma_t s_t + (1-\gamma_t) z_t$

end

5.1 Frank-Wolfe algorithm

The Frank-Wolfe algorithm (Frank and Wolfe, 1956; Jaggi, 2013) is used to minimize a convex differentiable function f over a convex bounded set $\mathcal{D}$. It is an iterative first-order optimization method. At each iteration t, the convex function f is approximated by a linear function defined by its gradient at the current point z_t. Then it finds the point s_t that minimizes that linear function, over the convex set $\mathcal{D}$:

$$s_t = \min_s s^\top \nabla f(z_t) \text{ s.t. } s \in \mathcal{D}.$$

The point z_{t+1} is then defined as the weighted average between the solution s_t and the current point z_t: $z_{t+1} = \gamma_t\, s_t + (1-\gamma_t)\, z_t$, where γ_t is the step size (such as $2/(t+2)$). Compared to the gradient descent algorithm, the Frank-Wolfe alogrithm does not take a step in the direction of the gradient, but in the direction of the point that minimizes the linear approximation of the function f over the convex set $\mathcal{D}$ (see Fig 3). In particular, this ensures that the points z_t always stay inside the convex set, and there is thus no need for a projection step.

To summarize, in order to use the Frank-Wolfe algorithm, we need to compute the gradient of the objective function and to minimize a linear function over our convex set. This is particularly appropriate to our problem, since we can easily minimize a linear function over the tree polytopes (using the minimum weight spanning tree algorithm), while projecting on those polytopes is more expensive.

Algorithm 2: Optimization algorithm for our method.

for $t \in \{1, ..., T\}$ **do**

Compute the optimal $\mathbf{w}$:

$$\mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2n}\|\mathbf{y}_t - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Compute the gradient w.r.t. $\mathbf{y}$:

$$\mathbf{g}_t = \frac{1}{n}\left(\mathbf{y}_t - \mathbf{X}\mathbf{w}_t\right) - \mu\ \mathbf{u}$$

Solve the linear program:

$$\mathbf{s}_t = \min_{\mathbf{s}\in\mathcal{Y}} \mathbf{s}^\top \mathbf{g}_t$$

Take the Frank-Wolfe step:

$$\mathbf{y}_{t+1} = \gamma_t \mathbf{s}_t + (1 - \gamma_t)\mathbf{y}_t$$

end

5.2 Application to our problem

We now describe how to use the Frank-Wolfe algorithm to optimize our objective function with respect to $\mathbf{y}$. First, let us introduce the functions f and h defined by

$$f(\mathbf{w}, \mathbf{y}) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 - \mu\ \mathbf{u}^\top \mathbf{y},$$
$$h(\mathbf{y}) = \min_{\mathbf{w}} f(\mathbf{w}, \mathbf{y}).$$

The original problem is equivalent to

$$\min_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{w}} f(\mathbf{w}, \mathbf{y}) = \min_{\mathbf{y}\in\mathcal{Y}} h(\mathbf{y}).$$

We will use the Frank-Wolfe algorithm to optimize the function h.

Minimizing w.r.t $\mathbf{w}$. First, we need to minimize the function f with respect to $\mathbf{w}$, in order to compute the function h (and its gradient). One must note that this is an unconstrained quadratic program, whose solution can be obtained in closed form by solving the linear system:

$$\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{w} = \mathbf{X}^\top \mathbf{y}.$$

However, in case of a very large feature space, this system might be prohibitively expensive to solve exactly. We instead propose to approximately compute the optimal $\mathbf{w}$ using stochastic gradient descent.

Computing the gradient of h. Then, the gradient of the function h at the point $\mathbf{y}$ is equal to

$$\nabla h(\mathbf{y}) = \nabla_{\mathbf{y}} f(\mathbf{w}^*, \mathbf{y}),$$

$\textsc{Pos}_i \times d$
$\textsc{Pos}_j \times d$
$\textsc{Pos}_i \times \textsc{Pos}_j \times d$
$\textsc{Pos}_i \times \textsc{Pos}_{i-1} \times \textsc{Pos}_j \times d$
$\textsc{Pos}_i \times \textsc{Pos}_{i+1} \times \textsc{Pos}_j \times d$
$\textsc{Pos}_i \times \textsc{Pos}_j \times \textsc{Pos}_{j-1} \times d$
$\textsc{Pos}_i \times \textsc{Pos}_j \times \textsc{Pos}_{j+1} \times d$

Table 2: Features used in our parser to describe the dependency between tokens i and j, where i is the head, j the dependent and $d = i - j$.

where $\mathbf{w}^*$ is equal to

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} f(\mathbf{w}, \mathbf{y}).$$

Thus, in order to compute the gradient of h with respect to $\mathbf{y}$, we start by computing the corresponding optimal value of $\mathbf{w}$. Then, the gradient with respect to $\mathbf{y}$ is equal to

$$\nabla h(\mathbf{y}) = \frac{1}{n}\left(\mathbf{y} - \mathbf{X}\mathbf{w}^*\right) - \mu\ \mathbf{u}.$$

Minimizing a linear function over $\mathcal{Y}$. We finaly need to compute the optimal solution of the following linear problem

$$\min_{\mathbf{s}\in\mathcal{Y}} \nabla h(\mathbf{y})^\top \mathbf{s}.$$

The optimal value of a linear function over a bounded convex polytope is always attained on at least one vertex of that polytope. By definition of our polytope, those vertices correspond to spanning trees. Thus, computing an optimal solution of this problem is obtained by finding a minimum weight spanning tree.

Discussion. Similarly to the Expectation-Maximization algorithm, our optimization method is a two-steps iterative algorithm. In the first step, the optimal parameter vector $\mathbf{w}$ is estimated based on the previous dependency trees, while the second step consist in re-estimating the (relaxed) dependency trees.

6 Experiments

In this section, we report the results of the experiments we have performed to evaluate our approach to grammar induction.

	DMV	PR	USR	OUR
DE	42.6	58.4	53.4	**60.2**
EN	22.4	57.5	**66.2**	62.3
ES	31.8	57.3	**71.5**	68.8
FR	56.0	66.2	54.1	**72.3**
ID	44.9	21.4	50.3	**69.7**
IT	33.3	40.4	46.5	**64.3**
JA	48.0	**58.9**	58.2	57.5
KO	35.3	50.7	48.8	**59.0**
PT-BR	49.6	40.7	46.4	**68.3**
SV	38.9	61.2	64.3	**66.2**
AVG	40.2	51.3	56.0	**64.8**

Table 3: Directed dependency accuracy, on the universal treebanks with universal parts-of-speech, on sentences of length 10 or less. PR refers to posterior regularization, USR to universal rules.

6.1 Features

The features used in our unsupervised parser are based on the parts-of-speech of the head and the dependent of the corresponding dependency, and are given in Table 2. Following McDonald et al. (2005a), we also include features capturing the context of the head or the dependent. These features are trigrams and are formed by the parts-of-speech of the two tokens of the dependency and one of the word appearing before/after the head/dependent. Finally, all the features are conjoined with the signed distance between the two words of the dependency.

6.2 Dataset

We use the universal treebanks, version 2.0, introduced by McDonald et al. (2013). This dataset contains dependency trees for ten languages belonging to five different families: Spanish, French, Italian, Portuguese (Romanic family), English, German, Swedish (Germanic family), Korean, Japanese and Indonesian. The tokens of those treebanks are tagged using the universal part-of-speech tagset (Petrov et al., 2012). We focus on inducing dependency grammars using universal parts-of-speech, and will thus report results where all methods use (gold) universal POS.

6.3 Comparison with baselines

We will compare our approach to three other unsupervised parsers. Our first baseline is the DMV model, introduced by Klein and Manning (2004).

DMV	PR	USR	OUR
7 min	1 h	15 h	2 min

Table 4: Computational times required to learn a grammar on the English treebank.

Our second baseline is the extended valence grammar model, with posterior sparsity constraints, as described by Gillenwater et al. (2011). Finally, our last baseline is the model with universal rules introduced by Naseem et al. (2010). It should be noted that these two baselines obtain performances that are near state-of-the-art. All methods are trained and tested on sentences of length 10 or less, after stripping punctuation.

Parameter selection. All the parameters were chosen using the English development set. Our method has two parameters, determined as: $\lambda = 0.001$ and $\mu = 0.1$. We used $T = 200$ iterations in all the experiments.

Discussion. We report the results in Table 3. First, we observe that our method performs better than the three baselines on seven out of ten languages. Overall, our approach outperforms the three baselines, with an absolute improvement of 13 points over the extended valence grammar with posterior sparsity and 8 points over the model with universal syntactic rules. We also note that the inter-language variance is lower for our method than the baselines (std of 4.6 for our method v.s. 8.3 for USR and 12.7 for PR). For the sake of completeness, we also compared those methods using the fine grained POS available in the universal treebanks. Overall, our method obtains an accuracy of 68.4, while USR and PR achieve accuracies of 67.3 and 58.5 respectively. Finally, we report computational times in Table 4, showing that our approach is much faster than the baselines.

6.4 Non-projective grammar induction

In this section, we investigate non-projective grammar induction. With our approach, we only have to replace the Eisner algorithm by Chu-Liu/Edmonds. We report results in Table 5. First, we observe that the non-projective results are slightly worse than projective one. This is not really surprising since the amount of non-projective gold dependencies is very small on the considered data. Moreover, non-projective trees are much more ambiguous than projective ones, leading to

	PROJECTIVE	NON-PROJECTIVE
DE	60.2	57.2
EN	62.3	60.5
ES	68.8	66.5
FR	72.3	69.2
ID	69.7	68.4
IT	64.3	63.1
JA	57.5	59.3
KO	59.0	60.0
PT-BR	68.3	67.7
SV	66.2	65.4
AVG	64.8	63.7

Table 5: Comparison between projective and non-projective unsupervised dependency parsing using our method.

a harder problem. We still believe those results are interesting because the difference is small (less than 1.5 points), while non-projective parsing is computationaly more efficient.

6.5 Evaluation on longer sentences

We also evaluate our method on longer sentences (while still training on sentences of length 10 or less). Directed dependency accuracies are reported in Figure 4. On all sentences, our method achieve an overall accuracy of 55.8.

6.6 Feature ablation study

In this section, we study the importance of the different features used in our parser. We report directed accuracies when different groups of features are removed, one at a time, in Table 6. First, we remove the distance information from the features (line DISTANCE). We observe that the performance of our parser is greatly affected by this ablation, especially for long sentences. Then, we remove the context features (line CONTEXT) and the unigram features (line UNIGRAM) from our model. We observe that the performance decreases slightly due to this ablations, but the differences are small.

	$\|\mathbf{w}\| \leq 10$	$\|\mathbf{w}\| \leq \infty$
DISTANCE	61.8	48.7
CONTEXT	64.2	55.1
UNIGRAM	64.0	55.3
ALL FEATURES	**64.8**	**55.8**

Table 6: Feature ablation study.

7 Discussion

In this paper, we introduced a new framework for the task of unsupervised dependency parsing. Our method is a based on a feature-rich discriminative model, whose parameters are learned using a convex objective function. We demonstrated on the universal treebanks that our approach leads to competitive results, while being computationaly very efficient. We now describe some directions we would like to explore as future work.

Richer feature set. In our experiments, we focused on assessing the usefulness of our convex, discriminative approach, and thus considered only relatively simple features based on parts-of-speech. Inspired by supervised dependency parsing, we would like to explore the use of other features such as Brown clusters (Brown et al., 1992) or distributed word representations (Mikolov et al., 2013), in order to lexicalize our parser.

Higher-order parsing. So far, our model is lacking the notion of valency, that has proven very useful for grammar induction. In future work, we would thus like to replace our edge-based factorization by a higher-order one, in order to capture siblings (and grandchilds) interactions. We would then have to use a higher-order parser, such as the ones described by McDonald and Pereira (2006) and Koo and Collins (2010). Another potential approach would be to use the linear programming relaxed inference, described by Martins et al. (2009).

Transfer learning. In this paper, we used universal syntactic rules, as described by Naseem et al. (2010) to guide our parser. We would like to explore the use of weak supervision, such as the one considered in transfer learning (Hwa et al., 2005). For example, projected dependencies from a resource-rich language could be used as constraints in our framework.

Code. The code for our method is distributed on the first author webpage.

Acknowledgments

This work is supported by National Science Foundation award 1344668 and National Institute of General Medical Sciences award R01 GM090187.

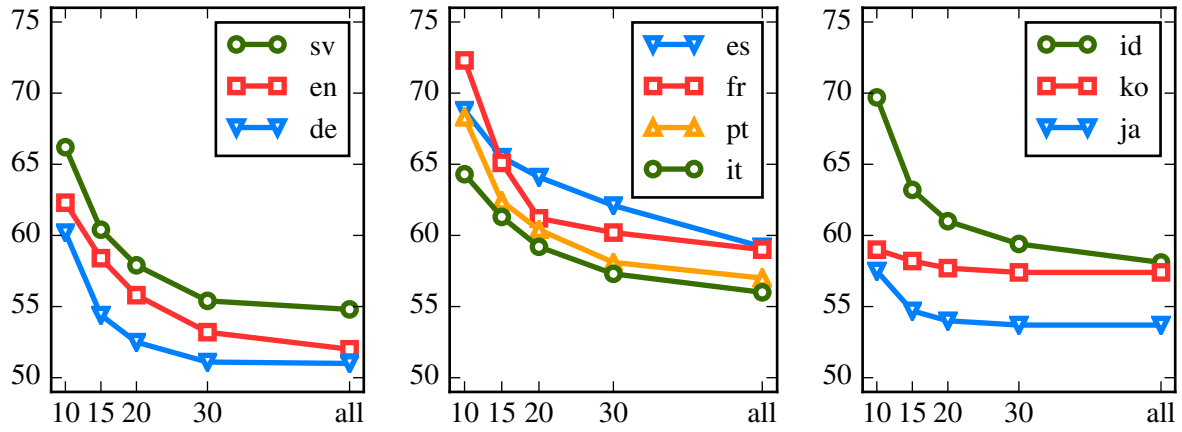

Figure 4: Directed dependency accuracies on longer sentences for our approach.

References

Francis R Bach and Zaïd Harchaoui. 2007. Diffrac: a discriminative and flexible framework for clustering. In *NIPS*.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *ACL*.

Yonatan Bisk and Julia Hockenmaier. 2013. An hdp model for inducing combinatory categorial grammars. *TACL*.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *EMNLP*.

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. 2014. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*.

Piotr Bojanowski, Rémi Lagugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. 2015. Weakly-supervised alignment of video with text. http://arxiv.org/abs/1505.06027.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*.

John Cocke and Jacob Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*.

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

Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval research logistics quarterly*.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *JMLR*.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. 2011. Posterior sparsity in unsupervised dependency parsing. *JMLR*.

Kevin Gimpel and Noah A Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *NAACL*.

Matthew R Gormley and Jason Eisner. 2013. Nonconvex global optimization for latent-variable models. In *ACL*.

Edouard Grave. 2014. A convex relaxation for weakly supervised relation extraction. In *EMNLP*.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *NAACL*.

Daniel Hsu, Percy Liang, and Sham M Kakade. 2012. Identifiability and unmixing of latent parse trees. In *NIPS*.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*.

Martin Jaggi. 2013. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*.

Armand Joulin, Jean Ponce, and Francis R Bach. 2010. Efficient optimization for discriminative latent class models. In *NIPS*.

Armand Joulin, Kevin Tang, and Li Fei-Fei. 2014. Efficient image and video co-localization with frank-wolfe algorithm. In *ECCV*.

Tadao Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report.

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

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*.

David Mareček and Zdeněk Žabokrtskỳ. 2012. Exploiting reducibility in unsupervised dependency parsing. In *EMNLP/CoNLL*.

André FT Martins, Noah A Smith, and Eric P Xing. 2009. Polyhedral outer approximations with application to natural language parsing. In *ICML*.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *EMNLP*.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *EMNLP*.

Ankur P Parikh, Shay B Cohen, and Eric P Xing. 2014. Spectral unsupervised parsing with additive tree metrics. In *ACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. *LREC*.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people with "their" names using coreference resolution. In *ECCV*.

Noah A Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *NAACL*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *CoNLL*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *EMNLP*.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks*.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. 2004. Maximum margin clustering. In *NIPS*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and control*.