MIT Open Access Articles

Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees

The MIT Faculty has made this article openly available. ***Please share*** how this access benefits you. Your story matters.

Citation	Zhang, Yuan, Tao Lei, Regina Barzilay, Tommi Jaakkola, and Amir Globerson. "Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees." 52nd Annual Meeting of the Association for Computational Linguistics (June 2014).
As Published	http://acl2014.org/acl2014/P14-1/index.html
Publisher	Association for Computational Linguistics
Version	Author's final manuscript
Accessed	Mon Jan 07 03:08:10 EST 2019
Citable Link	http://hdl.handle.net/1721.1/99746
Terms of Use	Creative Commons Attribution-Noncommercial-Share Alike
Detailed Terms	http://creativecommons.org/licenses/by-nc-sa/4.0/

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DSpace@MIT

Steps to Excellence: Simple Inference with Refined Scoring of Dependency Trees

Yuan Zhang, Tao Lei, Regina Barzilay, Tommi Jaakkola
Massachusetts Institute of Technology
{yuanzh, taolei, regina, tommi}@csail.mit.edu

Amir Globerson
The Hebrew University
gamir@cs.huji.ac.il

Abstract

Much of the recent work on dependency parsing has been focused on solving inherent combinatorial problems associated with rich scoring functions. In contrast, we demonstrate that highly expressive scoring functions can be used with substantially simpler inference procedures. Specifically, we introduce a sampling-based parser that can easily handle arbitrary global features. Inspired by SampleRank, we learn to take guided stochastic steps towards a high scoring parse. We introduce two samplers for traversing the space of trees, Gibbs and Metropolis-Hastings with Random Walk. The model outperforms state-of-the-art results when evaluated on 14 languages of non-projective CoNLL datasets. Our sampling-based approach naturally extends to joint prediction scenarios, such as joint parsing and POS correction. The resulting method outperforms the best reported results on the CATiB dataset, approaching performance of parsing with gold tags.[1]

1 Introduction

Dependency parsing is commonly cast as a maximization problem over a parameterized scoring function. In this view, the use of more expressive scoring functions leads to more challenging combinatorial problems of finding the maximizing parse. Much of the recent work on parsing has been focused on improving methods for solving the combinatorial maximization inference problems. Indeed, state-of-the-art results have been obtained by adapting powerful tools from optimization (Martins et al., 2013; Martins et al., 2011; Rush and Petrov, 2012). We depart from this view and instead focus on using highly expressive scoring functions with substantially simpler inference procedures. The key ingredient in our approach is how learning is coupled with inference. Our combination outperforms the state-of-the-art parsers and remains comparable even if we adopt their scoring functions.

Rich scoring functions have been used for some time. They first appeared in the context of reranking (Collins, 2000), where a simple parser is used to generate a candidate list which is then reranked according to the scoring function. Because the number of alternatives is small, the scoring function could in principle involve arbitrary (global) features of parse trees. The power of this methodology is nevertheless limited by the initial set of alternatives from the simpler parser. Indeed, the set may already omit the gold parse. We dispense with the notion of a candidate set and seek to exploit the scoring function more directly.

In this paper, we introduce a sampling-based parser that places few or no constraints on the scoring function. Starting with an initial candidate tree, our inference procedure climbs the scoring function in small (cheap) stochastic steps towards a high scoring parse. The proposal distribution over the moves is derived from the scoring function itself. Because the steps are small, the complexity of the scoring function has limited impact on the computational cost of the procedure. We explore two alternative proposal distributions. Our first strategy is akin to Gibbs sampling and samples a new head for each word in the sentence, modifying one arc at a time. The second strategy relies on a provably correct sampler for first-order scores (Wilson, 1996), and uses it within a Metropolis-Hastings algorithm for general scoring functions. It turns out that the latter optimizes the

[1] The source code for the work is available at http://groups.csail.mit.edu/rbg/code/global/acl2014.

score more efficiently than the former.

Because the inference procedure is so simple, it is important that the parameters of the scoring function are chosen in a manner that facilitates how we climb the scoring function in small steps. One way to achieve this is to make sure that improvements in the scoring functions are correlated with improvements in the quality of the parse. This approach was suggested in the SampleRank framework (Wick et al., 2011) for training structured prediction models. This method was originally developed for a sequence labeling task with local features, and was shown to be more effective than state-of-the-art alternatives. Here we apply SampleRank to parsing, applying several modifications such as the proposal distributions mentioned earlier.

The benefits of sampling-based learning go beyond stand-alone parsing. For instance, we can use the framework to correct preprocessing mistakes in features such as part-of-speech (POS) tags. In this case, we combine the scoring function for trees with a stand-alone tagging model. When proposing a small move, i.e., sampling a head of the word, we can also jointly sample its POS tag from a set of alternatives provided by the tagger. As a result, the selected tag is influenced by a broad syntactic context above and beyond the initial tagging model and is directly optimized to improve parsing performance. Our joint parsing-tagging model provides an alternative to the widely-adopted pipeline setup.

We evaluate our method on benchmark multilingual dependency corpora. Our method outperforms the Turbo parser across 14 languages on average by 0.5%. On four languages, we top the best published results. Our method provides a more effective mechanism for handling global features than reranking, outperforming it by 1.3%. In terms of joint parsing and tagging on the CATiB dataset, we nearly bridge (88.38%) the gap between independently predicted (86.95%) and gold tags (88.45%). This is better than the best published results in the 2013 SPMRL shared task (Seddah et al., 2013), including parser ensembles.

2 Related Work

Earlier works on dependency parsing focused on inference with tractable scoring functions. For instance, a scoring function that operates over each single dependency can be optimized using the maximum spanning tree algorithm (McDonald et al., 2005). It was soon realized that using higher order features could be beneficial, even at the cost of using approximate inference and sacrificing optimality. The first successful approach in this arena was reranking (Collins, 2000; Charniak and Johnson, 2005) on constituency parsing. Reranking can be combined with an arbitrary scoring function, and thus can easily incorporate global features over the entire parse tree. Its main disadvantage is that the output parse can only be one of the few parses passed to the reranker.

Recent work has focused on more powerful inference mechanisms that consider the full search space (Zhang and McDonald, 2012; Rush and Petrov, 2012; Koo et al., 2010; Huang, 2008). For instance, Nakagawa (2007) deals with tractability issues by using sampling to approximate marginals. Another example is the dual decomposition (DD) framework (Koo et al., 2010; Martins et al., 2011). The idea in DD is to decompose the hard maximization problem into smaller parts that can be efficiently maximized and enforce agreement among these via Lagrange multipliers. The method is essentially equivalent to linear programming relaxation approaches (Martins et al., 2009; Sontag et al., 2011), and also similar in spirit to ILP approaches (Punyakanok et al., 2004).

A natural approach to approximate global inference is via search. For instance, a transition-based parsing system (Zhang and Nivre, 2011) incrementally constructs a parsing structure using greedy beam-search. Other approaches operate over full trees and generate a sequence of candidates that successively increase the score (Daumé III et al., 2009; Li et al., 2013; Wick et al., 2011). Our work builds on one such approach — SampleRank (Wick et al., 2011), a sampling-based learning algorithm. In SampleRank, the parameters are adjusted so as to guide the sequence of candidates closer to the target structure along the search path. The method has been successfully used in sequence labeling and machine translation (Haddow et al., 2011). In this paper, we demonstrate how to adapt the method for parsing with rich scoring functions.

3 Sampling-Based Dependency Parsing with Global Features

In this section, we introduce our novel sampling-based dependency parser which can incorporate

arbitrary global features. We begin with the notation before addressing the decoding and learning algorithms. Finally, we extend our model to a joint parsing and POS correction task.

3.1 Notations

We denote sentences by x and the corresponding dependency trees by $y \in \mathcal{Y}(x)$. Here $\mathcal{Y}(x)$ is the set of valid (projective or non-projective) dependency trees for sentence x. We use x_j to refer to the jth word of sentence x, and h_j to the head word of x_j. A training set of size N is given as a set of pairs $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ where $y^{(i)}$ is the ground truth parse for sentence $x^{(i)}$.

We parameterize the scoring function $s(x, y)$ as

$$s(x, y) = \theta \cdot f(x, y) \tag{1}$$

where $f(x, y)$ is the feature vector associated with tree y for sentence x. We do not make any assumptions about how the feature function decomposes. In contrast, most state-of-the-art parsers operate under the assumption that the feature function decomposes into a sum of simpler terms. For example, in the second-order MST parser (McDonald and Pereira, 2006), all the feature terms involve arcs or consecutive siblings. Similarly, parsers based on dual decomposition (Martins et al., 2011; Koo et al., 2010) assume that $s(x, y)$ decomposes into a sum of terms where each term can be maximized over y efficiently.

3.2 Decoding

The decoding problem consists of finding a valid dependency tree $y \in \mathcal{Y}(x)$ that maximizes the score $s(x, y) = \theta \cdot f(x, y)$ with parameters θ. For scoring functions that extend beyond first-order arc preferences, finding the maximizing non-projective tree is known to be NP-hard (McDonald and Pereira, 2006). We find a high scoring tree through sampling, and (later) learn the parameters θ so as to further guide this process.

Our sampler generates a sequence of dependency structures so as to approximate independent samples from

$$p(y|x, T, \theta) \propto \exp\left(s(x, y)/T\right) \tag{2}$$

The temperature parameter T controls how concentrated the samples are around the maximum of $s(x, y)$ (e.g., see Geman and Geman (1984)). Sampling from target distribution p is typically as hard as (or harder than) that maximizing $s(x, y)$.

Inputs: θ, x, T_0 (initial temperature), c (temperature update rate), proposal distribution q.
Outputs: y^*
$T \leftarrow T_0$
Set y^0 to some random tree
$y^* \leftarrow y^0$
repeat
  $y' \leftarrow q(\cdot|x, y^t, T, \theta)$
  if $s(x, y') > s(x, y^*)$ **then**
    $y^* \leftarrow y'$
  $\alpha = \min\left[1, \frac{p(y')q(y^t|y')}{p(y^t)q(y'|y^t)}\right]$
  Sample Bernouli variable Z with $P[Z = 1] = \alpha$.
  if $Z = 0$ **then**
    $y^{t+1} \leftarrow y^t$
  else
    $y^{t+1} \leftarrow y'$
  $t \leftarrow t + 1$
  $T \leftarrow c \cdot T$
until convergence
return y^*

Figure 1: Sampling-based algorithm for decoding (i.e., approximately maximizing $s(x, y)$).

We follow here a Metropolis-Hastings sampling algorithm (e.g., see Andrieu et al. (2003)) and explore different alternative proposal distributions $q(y'|x, y, \theta, T)$. The distribution q governs the small steps that are taken in generating a sequence of structures. The target distribution p folds into the procedure by defining the probability that we will accept the proposed move. The general structure of our sampling algorithm is given in Figure 1.

3.2.1 Gibbs Sampling

Perhaps the most natural choice of the proposal distribution q is a conditional distribution from p. This is feasible if we restrict the proposed moves to only small changes in the current tree. In our case, we choose a word j randomly, and then sample its head h_j according to p with the constraint that we obtain a valid tree (when projective trees are sought, this constraint is also incorporated). For this choice of q, the probability of accepting the new tree (α in Figure 1) is identically one. Thus new moves are always accepted.

3.2.2 Exact First-Order Sampling

One shortcoming of the Gibbs sampler is that it only changes one variable (arc) at a time. This usually leads to slow mixing, requiring more samples to get close to the parse with maximum score. Ideally, we would change multiple heads in the parse tree simultaneously, and sample those choices from the corresponding conditional distribution of p. While in general this is increasingly difficult with more heads, it is indeed tractable if

Inputs: x, y^t, θ, K (number of heads to change).
Outputs: y'
for $i = 1$ **to** $|x|$ **do**
  $inTree[i] \leftarrow false$
  $ChangeNode[i] \leftarrow false$
Set $ChangeNode$ to true for K random nodes.
$head[0] \leftarrow -1$
for $i = 1$ **to** $|x|$ **do**
  $u \leftarrow i$
  while not $inTree[u]$ **do**
    if $ChangeNode[u]$ **then**
      $head[u] \leftarrow$ randomHead(u, θ)
    else
      $head[u] \leftarrow y^t(u)$
    $u \leftarrow head[u]$
  $u \leftarrow i$
  while not $inTree[u]$ **do**
    $inTree[u] \leftarrow true$
    $u \leftarrow head[u]$
return Construct tree y' from the head array.

Figure 2: A proposal distribution $q(y'|y^t)$ based on the random walk sampler of Wilson (1996). The function randomHead samples a new head for node u according to the first-order weights given by θ.

the model corresponds to a first-order parser. One such sampling algorithm is the random walk sampler of Wilson (1996). It can be used to obtain i.i.d. samples from distributions of the form:

$$p(y) \propto \prod_{i \to j \in y} w_{ij}, \qquad (3)$$

where y corresponds to a tree with a spcified root and w_{ij} is the exponential of the first-order score. y is always a valid parse tree if we allow multiple children of the root and do not impose projective constraint. The algorithm in Wilson (1996) iterates over all the nodes, and for each node performs a random walk according to the weights w_{ij} until the walk creates a loop or hits a tree. In the first case the algorithm can erase the loop by continuing the walk. If the walk hits the current tree, the walk path is added to form a new tree with more nodes. This is repeated until all the nodes are included in the tree. It can be shown that this procedure generates i.i.d. trees from $p(y)$.

Since our features do not by design correspond to a first-order parser, we cannot use the Wilson algorithm as it is. Instead we use it as the proposal function and sample a subset of the dependencies from the first-order distribution of our model, while fixing the others. In each step we uniformly sample K nodes to update and sample their new heads using the Wilson algorithm (in the experiments we use $K = 4$). Note that blocked Gibbs sampling would be exponential in K, and is thus very slow already at $K = 4$. The procedure is described in Figure 2 with a graphic illustration in Figure 3.

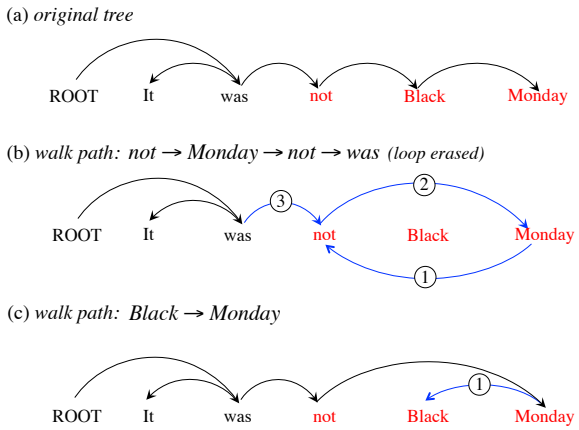


Figure 3: An illustration of random walk sampler. The index on each edge indicates its order on each walk path. The heads of the red words are sampled while others are fixed. The blue edges represent the current walk path and the black ones are already in the tree. Note that the walk direction is opposite to the dependency direction. (a) shows the original tree before sampling; (b) and (c) show the walk path and how the tree is generated in two steps. The loop *not*→ *Monday* → *not* in (b) is erased.

3.3 Training

In this section, we describe how to learn the adjustable parameters θ in the scoring function. The parameters are learned in an on-line fashion by successively imposing soft constraints between pairs of dependency structures. We introduce both margin constraints and constraints pertaining to successive samples generated along the search path. We demonstrate later that both types of constraints are essential.

We begin with the standard margin constraints. An ideal scoring function would always rank the gold parse higher than any alternative. Moreover, alternatives that are far from the gold parse should score even lower. As a result, we require that

$$s(x^{(i)}, y^{(i)}) - s(x^{(i)}, y) \geq \Delta(y^{(i)}, y) \quad \forall y \qquad (4)$$

where $\Delta(y^{(i)}, y)$ is the number of head mistakes in y relative to the gold parse $y^{(i)}$. We adopt here a shorthand $Err(y) = \Delta(y^{(i)}, y)$, where the dependence on $y^{(i)}$ is implied from context. Note

that Equation 4 contains exponentially many constraints and cannot be enforced jointly for general scoring functions. However, our sampling procedure generates a small number of structures along the search path. We enforce only constraints corresponding to those samples.

The second type of constraints are enforced between successive samples along the search path. To illustrate the idea, consider a parse y that differs from $y^{(i)}$ in only one arc, and a parse y' that differs from $y^{(i)}$ in ten arcs. We cannot necessarily assume that $s(x, y)$ is greater than $s(x, y')$ without additional encouragement. Thus, we can complement the constraints in Equation 4 with additional pairwise constraints (Wick et al., 2011):

$$s(x^{(i)}, y) - s(x^{(i)}, y') \geq Err(y') - Err(y) \quad (5)$$

where similarly to Equation 4, the difference in scores scales with the differences in errors with respect to the target $y^{(i)}$. We only enforce the above constraints for y, y' that are consecutive samples in the course of the sampling process. These constraints serve to guide the sampling process derived from the scoring function towards the gold parse.

We learn the parameters θ in an on-line fashion to satisfy the above constraints. This is done via the MIRA algorithm (Crammer and Singer, 2003). Specifically, if the current parameters are θ_t, and we enforce constraint Equation 5 for a particular pair y, y', then we will find θ_{t+1} that minimizes

$$\begin{array}{ll} \min & ||\theta - \theta_t||^2 + C\xi \\ \text{s.}t. & \theta \cdot (f(x, y) - f(x, y')) \geq Err(y') - Err(y) - \xi \end{array} \quad (6)$$

The updates can be calculated in closed form. Figure 4 summarizes the learning algorithm. We repeatedly generate parses based on the current parameters θ_t for each sentence $x^{(i)}$, and use successive samples to enforce constraints in Equation 4 and Equation 5 one at a time.

3.4 Joint Parsing and POS Correction

It is easy to extend our sampling-based parsing framework to joint prediction of parsing and other labels. Specifically, when sampling the new heads, we can also sample the values of other variables at the same time. For instance, we can sample the POS tag, the dependency relation or morphology information. In this work, we investigate a joint POS correction scenario in which only the predicted POS tags are provided in the testing phase, while both gold and predicted tags are available for the training set.

Inputs: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$.
Outputs: Learned parameters θ.
$\theta_0 \leftarrow \mathbf{0}$
for $e = 1$ **to** #epochs **do**
  for $i = 1$ **to** N **do**
    $y' \leftarrow q(\cdot|x^{(i)}, y_i^{t_i}, \theta_t)$
    $y^+ = \arg\min_{y \in \left\{y_i^{t_i}, y'\right\}} Err(y)$
    $y^- = \arg\max_{y \in \left\{y_i^{t_i}, y'\right\}} Err(y)$
    $y_i^{t_i+1} \leftarrow$ acceptOrReject($y', y_i^{t_i}, \theta_t$)
    $t_i \leftarrow t_i + 1$
    $\nabla f = f(x^{(i)}, y^+) - f(x^{(i)}, y^-)$
    $\Delta Err = Err(y^+) - Err(y^-)$
    if $\Delta Err \neq 0$ and $\theta_t \cdot \nabla f < \Delta Err$ **then**
      $\theta_{t+1} \leftarrow$ updateMIRA($\nabla f, \Delta Err, \theta_t$)
      $t \leftarrow t + 1$
    $\nabla f_g = f(x^{(i)}, y^{(i)}) - f(x^{(i)}, y_i^{t_i})$
    if $\theta_t \cdot \nabla f_g < Err(y_i^{t_i})$ **then**
      $\theta_{t+1} \leftarrow$ updateMIRA($\nabla f_g, Err(y_i^{t_i}), \theta_t$)
      $t \leftarrow t + 1$
return Average of $\theta_0, \ldots, \theta_t$ parameters.

Figure 4: SampleRank algorithm for learning. The rejection strategy is as in Figure 1. $y_i^{t_i}$ is the t_ith tree sample of $x^{(i)}$. The first MIRA update (see Equation 6) enforces a ranking constraint between two sampled parses. The second MIRA update enforces constraints between a sampled parse and the gold parse. In practice several samples are drawn for each sentence in each epoch.

We extend our model such that it jointly learns how to predict a parse tree and also correct the predicted POS tags for a better parsing performance. We generate the POS candidate list for each word based on the confusion matrix on the training set. Let $c(t_g, t_p)$ be the count when the gold tag is t_g and the predicted one is t_p. For each word w, we first prune out its POS candidates by using the vocabulary from the training set. We don't prune anything if w is unseen. Assuming that the predicted tag for w is t_p, we further remove those tags t if their counts are smaller than some threshold $c(t, t_p) < \alpha \cdot c(t_p, t_p)$[2].

After generating the candidate lists for each word, the rest of the extension is rather straightforward. For each sampling, let $\mathcal{H}$ be the set of candidate heads and $\mathcal{T}$ be the set of candidate POS tags. The Gibbs sampler will generate a new sample from the space $\mathcal{H} \times \mathcal{T}$. The other parts of the algorithm remain the same.

[2]In our work we choose $\alpha = 0.003$, which gives a 98.9% oracle POS tagging accuracy on the CATiB development set.

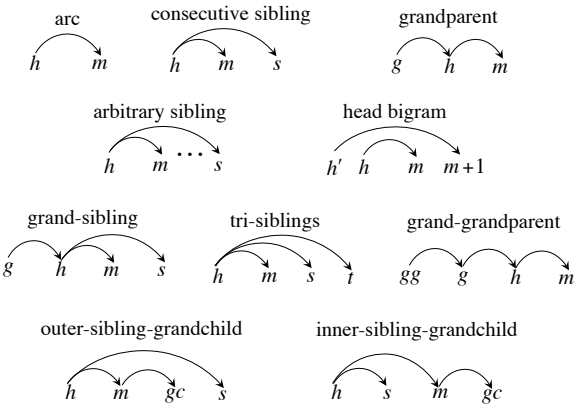


Figure 5: First- to third-order features.

4 Features

First- to Third-Order Features The feature templates of first- to third-order features are mainly drawn from previous work on graph-based parsing (McDonald and Pereira, 2006), transition-based parsing (Nivre et al., 2006) and dual decomposition-based parsing (Martins et al., 2011). As shown in Figure 5, the *arc* is the basic structure for first-order features. We also define features based on consecutive sibling, grandparent, arbitrary sibling, head bigram, grand-sibling and tri-siblings, which are also used in the Turbo parser (Martins et al., 2013). In addition to these first- to third-order structures, we also consider grand-grandparent and sibling-grandchild structures. There are two types of sibling-grandchild structures: (1) inner-sibling when the sibling is between the head and the modifier and (2) outer-sibling for the other cases.

Global Features We used feature shown promising in prior reranking work Charniak and Johnson (2005), Collins (2000) and Huang (2008).

- **Right Branch** This feature enables the model to prefer right or left-branching trees. It counts the number of words on the path from the root node to the right-most non-punctuation word, normalized by the length of the sentence.
- **Coordination** In a coordinate structure, the two adjacent conjuncts usually agree with each other on POS tags and their span lengths. For instance, in *cats and dogs*, the conjuncts are both short noun phrases. Therefore, we add different features to capture POS tag and span length consistency in a coordinate structure.
- **PP Attachment** We add features of lexical tuples involving the head, the argument and the preposition of prepositional phrases. Generally, this feature can be defined based on an instance of grandparent structure. However, we also handle the case of coordination. In this case, the arguments should be the conjuncts rather than the coordinator. Figure 6 shows an example.

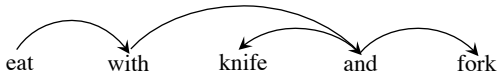


Figure 6: An example of PP attachment with coordination. The arguments should be *knife* and *fork*, not *and*.

- **Span Length** This feature captures the distribution of the binned span length of each POS tag. It also includes flags of whether the span reaches the end of the sentence and whether the span is followed by the punctuation.
- **Neighbors** The POS tags of the neighboring words to the left and right of each span, together with the binned span length and the POS tag at the span root.
- **Valency** We consider valency features for each POS tag. Specifically, we add two types of valency information: (1) the binned number of non-punctuation modifiers and (2) the concatenated POS string of all those modifiers.
- **Non-projective Arcs** A flag indicating if a dependency is projective or not (i.e. if it spans a word that does not descend from its head) (Martins et al., 2011). This flag is also combined with the POS tags or the lexical words of the head and the modifier.

POS Tag Features In the joint POS correction scenario, we also add additional features specifically for POS prediction. The feature templates are inspired by previous feature-rich POS tagging work (Toutanova et al., 2003). However, we are free to add higher order features because we do not rely on dynamic programming decoding. In our work we use feature templates up to 5-gram. Table 1 summarizes all POS tag feature templates.

5 Experimental Setup

Datasets We evaluate our model on standard benchmark corpora — CoNLL 2006 and CoNLL 2008 (Buchholz and Marsi, 2006; Surdeanu et al., 2008) — which include dependency treebanks for 14 different languages. Most of these data sets

1-gram	$\langle t_i \rangle, \langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle$
2-gram	$\langle t_{i-1}, t_i \rangle, \langle t_{i-2}, t_i \rangle, \langle t_{i-1}, t_i, w_{i-1} \rangle, \langle t_{i-1}, t_i, w_i \rangle$
3-gram	$\langle t_{i-1}, t_i, t_{i+1} \rangle, \langle t_{i-2}, t_i, t_{i+1}, \rangle, \langle t_{i-1}, t_i, t_{i+2} \rangle, \langle t_{i-2}, t_i, t_{i+2} \rangle$
4-gram	$\langle t_{i-2}, t_{i-1}, t_i, t_{i+1} \rangle, \langle t_{i-2}, t_{i-1}, t_i, t_{i+2} \rangle, \langle t_{i-2}, t_i, t_{i+1}, t_{i+2} \rangle$
5-gram	$\langle t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2} \rangle$

Table 1: POS tag feature templates. t_i and w_i denotes the POS tag and the word at the current position. t_{i-x} and t_{i+x} denote the left and right context tags, and similarly for words.

contain non-projective dependency trees. We use all sentences in CoNLL datasets during training and testing. We also use the Columbia Arabic Treebank (CATiB) (Marton et al., 2013). CATiB mostly includes projective trees. The trees are annotated with both gold and predicted versions of POS tags and morphology information. Following Marton et al. (2013), for this dataset we use 12 core POS tags, word lemmas, determiner features, rationality features and functional genders and numbers.

Some CATiB sentences exceed 200 tokens. For efficiency, we limit the sentence length to 70 tokens in training and development sets. However, we do not impose this constraint during testing. We handle long sentences during testing by applying a simple split-merge strategy. We split the sentence based on the ending punctuation, predict the parse tree for each segment and group the roots of resulting trees into a single node.

Evaluation Measures Following standard practice, we use Unlabeled Attachment Score (UAS) as the evaluation metric in all our experiments. We report UAS excluding punctuation on CoNLL datasets, following Martins et al. (2013). For the CATiB dataset, we report UAS including punctuation in order to be consistent with the published results in the 2013 SPMRL shared task (Seddah et al., 2013).

Baselines We compare our model with the Turbo parser and the MST parser. For the Turbo parser, we directly compare with the recent published results in (Martins et al., 2013). For the MST parser, we train a second-order non-projective model using the most recent version of the code[3].

We also compare our model against a discriminative reranker. The reranker operates over the top-50 list obtained from the MST parser[4]. We use a 10-fold cross-validation to generate candidate lists for training. We then train the reranker by running 10 epochs of cost-augmented MIRA. The reranker uses the same features as our model, along with the tree scores obtained from the MST parser (which is a standard practice in reranking).

Experimental Details Following Koo and Collins (2010), we always first train a first-order pruner. For each word x_i, we prune away the incoming dependencies $\langle h_i, x_i \rangle$ with probability less than 0.005 times the probability of the most likely head, and limit the number of candidate heads up to 30. This gives a 99% pruning recall on the CATiB development set. The first-order model is also trained using the algorithm in Figure 4. After pruning, we tune the regularization parameter $C = \{0.1, 0.01, 0.001\}$ on development sets for different languages. Because the CoNLL datasets do not have a standard development set, we randomly select a held out of 200 sentences from the training set. We also pick the training epochs from $\{50, 100, 150\}$ which gives the best performance on the development set for each language. After tuning, the model is trained on the full training set with the selected parameters.

We apply the Random Walk-based sampling method (see Section 3.2.2) for the standard dependency parsing task. However, for the joint parsing and POS correction on the CATiB dataset we do not use the Random Walk method because the first-order features in normal parsing are no longer first-order when POS tags are also variables. Therefore, the first-order distribution is not well-defined and we only employ Gibbs sampling for simplicity. On the CATiB dataset, we restrict the sample trees to always be projective as described in Section 3.2.1. However, we do not impose this constraint for the CoNLL datasets.

6 Results

Comparison with State-of-the-art Parsers Table 2 summarizes the performance of our model and of the baselines. We first compare our model to the Turbo parser using the Turbo parser feature set. This is meant to test how our learning and inference methods compare to a dual decomposition approach. The first column in Table 2

[3] http://sourceforge.net/projects/mstparser/

[4] The MST parser is trained in projective mode for reranking because generating top-k list from second-order non-projective model is intractable.

	Our Model (UAS)		Turbo (UAS)	MST 2nd-Ord. (UAS)	Best Published UAS	Top-50 Reranker	Top-500 Reranker
	Turbo Feat.	Full Feat.					
Arabic	79.86	80.21	79.64	78.75	81.12 (Ma11)	79.03	78.91
Bulgarian	92.97	93.30	93.10	91.56	94.02 (Zh13)	92.81	-
Chinese	92.06	**92.63**	89.98	91.77	91.89 (Ma10)	92.25	-
Czech	90.62	**91.04**	90.32	87.30	90.32 (Ma13)	88.14	-
Danish	91.45	91.80	91.48	90.50	92.00 (Zh13)	90.88	90.91
Dutch	85.83	**86.47**	86.19	84.11	86.19 (Ma13)	81.01	-
English	92.79	92.94	93.22	91.54	93.22 (Ma13)	92.41	-
German	91.79	92.07	92.41	90.14	92.41 (Ma13)	91.19	-
Japanese	93.23	93.42	93.52	92.92	93.72 (Ma11)	93.40	-
Portuguese	91.82	92.41	92.69	91.08	93.03 (Ko10)	91.47	-
Slovene	86.19	86.82	86.01	83.25	86.95 (Ma11)	84.81	85.37
Spanish	**88.24**	88.21	85.59	84.33	87.96 (Zh13)	86.85	87.21
Swedish	90.48	90.71	91.14	89.05	91.62 (Zh13)	90.53	-
Turkish	76.82	77.21	76.90	74.39	77.55 (Ko10)	76.35	76.23
Average	88.87	89.23	88.72	86.86	89.43	87.92	-

Table 2: Results of our model, the Turbo parser, and the MST parser. "Best Published UAS" includes the most accurate parsers among Nivre et al. (2006), McDonald et al. (2006), Martins et al. (2010), Martins et al. (2011), Martins et al. (2013), Koo et al. (2010), Rush and Petrov (2012), Zhang and McDonald (2012) and Zhang et al. (2013). Martins et al. (2013) is the current Turbo parser. The last two columns shows UAS of the discriminative reranker.

shows the result for our model with an average of 88.87%, and the third column shows the results for the Turbo parser with an average of 88.72%. This suggests that our learning and inference procedures are as effective as the dual decomposition method in the Turbo parser.

Next, we add global features that are not used by the Turbo parser. The performance of our model is shown in the second column with an average of 89.23%. It outperforms the Turbo parser by 0.5% and achieves the best reported performance on four languages. Moreover, our model also outperforms the 88.80% average UAS reported in Martins et al. (2011), which is the top performing single parsing system (to the best of our knowledge).

Comparison with Reranking As column 6 of Table 2 shows, our model outperforms the reranker by 1.3%[5]. One possible explanation of this performance gap between the reranker and our model is the small number of candidates considered by the reranker. To test this hypothesis, we performed experiments with top-500 list for a subset of languages.[6] As column 7 shows, this increase in the list size does not change the relative performance of the reranker and our model.

Joint Parsing and POS Correction Table 3 shows the results of joint parsing and POS correction on the CATiB dataset, for our model and state-of-the-art systems. As the upper part of the table shows, the parser with corrected tags reaches 88.38% compared to the accuracy of 88.46% on the gold tags. This is a substantial increase from the parser that uses predicted tags (86.95%).

To put these numbers into perspective, the bottom part of Table 3 shows the accuracy of the best systems from the 2013 SPMRL shared task on Arabic parsing using predicted information (Seddah et al., 2013). Our system not only outperforms the best single system (Björkelund et al., 2013) by 1.4%, but it also tops the ensemble system that combines three powerful parsers: the Mate parser (Bohnet, 2010), the Easy-First parser (Goldberg and Elhadad, 2010) and the Turbo parser (Martins et al., 2013)

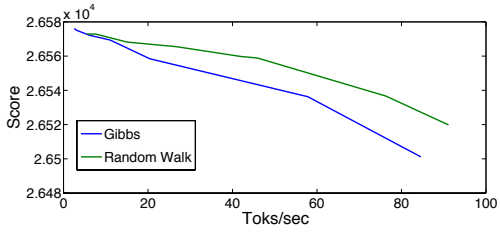
Impact of Sampling Methods We compare two sampling methods introduced in Section 3.2 with respect to their decoding efficiency. Specifically, we measure the score of the retrieved trees in testing as a function of the decoding speed, measured by the number of tokens per second. We change the temperature update rate c in order to decode with different speed. In Figure 7 we show the corresponding curves for two languages: Arabic and Chinese. We select these two languages as they correspond to two extremes in sentence length: Arabic has the longest sentences on average, while Chinese has the shortest ones. For both languages, the tree score improves over time. Given sufficient time, both sampling methods achieve the same score. However, the Random Walk-based sampler performs better when the quality is traded for speed. This result is to be expected given that each

[5]Note that the comparison is conservative because we can also add MST scores as features in our model as in reranker. With these features our model achieves an average UAS 89.28%.

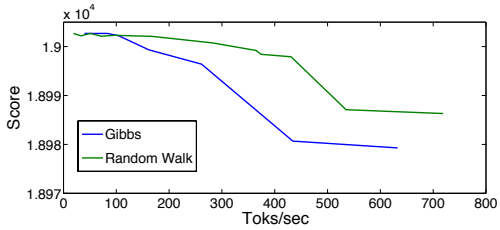
[6]We ran this experiment on 5 languages with small datasets due to the scalability issues associated with reranking top-500 list.

	Dev. Set ($\leq$ 70)		Testing Set	
	POS Acc.	UAS	POS Acc.	UAS
Gold	-	90.27	-	88.46
Predicted	96.87	88.81	96.82	86.95
POS Correction	97.72	90.08	97.49	**88.38**
CADIM	96.87	87.4-	96.82	85.78
IMS-Single	-	-	-	86.96
IMS-Ensemble	-	-	-	88.32

Table 3: Results for parsing and corrective tagging on the CATiB dataset. The upper part shows UAS of our model with gold/predicted information or POS correction. Bottom part shows UAS of the best systems in the SPMRL shared task. IMS-Single (Björkelund et al., 2013) is the best single parsing system, while IMS-Ensemble (Björkelund et al., 2013) is the best ensemble parsing system. We also show results for CADIM (Marton et al., 2013), the second best system, because we use their predicted features.



(a) Arabic



(b) Chinese

Figure 7: Total score of the predicted test trees as a function of the decoding speed, measured in the number of tokens per second.

iteration of this sampler makes multiple changes to the tree, in contrast to a single-edge change of Gibbs sampler.

The Effect of Constraints in Learning Our training method updates parameters to satisfy the pairwise constraints between (1) subsequent samples on the sampling path and (2) selected samples and the ground truth. Figure 8 shows that applying both types of constraints is consistently better than using either of them alone. Moreover, these results demonstrate that comparison between subsequent samples is more important than comparison against the gold tree.

Decoding Speed Our sampling-based parser is an anytime algorithm, and therefore its running time can be traded for performance. Figure 7 illustrates this trade-off. In the experiments reported above, we chose a conservative cooling rate and continued to sample until the score no longer changed. The parser still managed to process all the datasets in a reasonable time. For example, the time that it took to decode all the test sentences in Chinese and Arabic were 3min and 15min, respectively. Our current implementation is in Java and can be further optimized for speed.

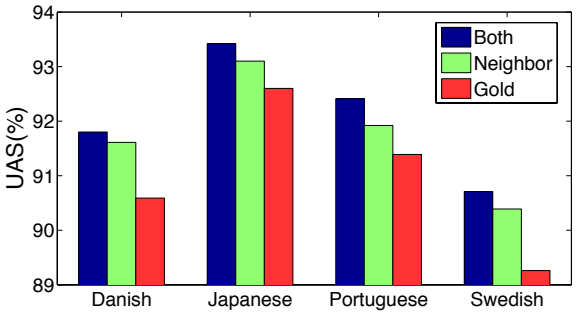


Figure 8: UAS on four languages when training with different constraints. "Neighbor" corresponds to pairwise constraints between subsequent samples, "Gold" represents constraints between a single sample and the ground truth, "Both" means applying both types of constraints.

7 Conclusions

This paper demonstrates the power of combining a simple inference procedure with a highly expressive scoring function. Our model achieves the best results on the standard dependency parsing benchmark, outperforming parsing methods with elaborate inference procedures. In addition, this framework provides simple and effective means for joint parsing and corrective tagging.

Acknowledgments

This research is developed in collaboration with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the IYAS project. The authors acknowledge the support of the MURI program (W911NF-10-1-0533, the DARPA BOLT program and the US-Israel Binational Science Foundation (BSF, Grant No 2012330). We thank the MIT NLP group and the ACL reviewers for their comments.

References

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. 2003. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *COLING*, pages 89–97.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Barry Haddow, Abhishek Arun, and Philipp Koehn. 2011. Samplerank training for phrase-based machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 261–271. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

Quannan Li, Jingdong Wang, Zhuowen Tu, and David P Wipf. 2013. Fixed-point model for structured labeling. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 214–221.

André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–249. Association for Computational Linguistics.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yuval Marton, Nizar Habash, Owen Rambow, and Sarah Alkhulani. 2013. Spmrl13 shared task system: The cadim arabic dependency parser. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–80.

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

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics.

Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *EMNLP-CoNLL*, pages 952–956.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics.

Alexander M Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 498–507. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, et al. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182.

D. Sontag, A. Globerson, and T. Jaakkola. 2011. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, pages 219–254. MIT Press.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Michael L. Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. Samplerank: Training factor graphs with atomic gradients. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 777–784.

David Bruce Wilson. 1996. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303. ACM.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics.

Hao Zhang, Liang Huang Kai Zhao, and Ryan McDonald. 2013. Online learning for inexact hypergraph search. In *Proceedings of EMNLP*.