Unsupervised Parsing with S-DIORA: Single Tree Encoding for Deep Inside-Outside Recursive Autoencoders

Andrew Drozdov, Subendhu Rongali, Yi-Pei Chen, Tim O'Gorman, Mohit Iyyer, and Andrew McCallum

College of Information and Computer Sciences
University of Massachusetts Amherst

{adrozdov, srongali, yipeichen, togorman, miyyer, mccallum}
@cs.umass.edu

Abstract

The deep inside-outside recursive autoencoder (DIORA; Drozdov et al. 2019a) is a self-supervised neural model that learns to induce syntactic tree structures for input sentences *without access to labeled training data*. In this paper, we discover that while DIORA exhaustively encodes all possible binary trees of a sentence with a soft dynamic program, its vector averaging approach is locally greedy and cannot recover from errors when computing the highest scoring parse tree in bottom-up chart parsing. To fix this issue, we introduce S-DIORA, an improved variant of DIORA that encodes a single tree rather than a softly-weighted mixture of trees by employing a hard argmax operation and a beam at each cell in the chart. Our experiments show that through *fine-tuning* a pre-trained DIORA with our new algorithm, we improve the state of the art in *unsupervised* constituency parsing on the English WSJ Penn Treebank by $\mathbf{2.2 - 6}\%$ F1, depending on the data used for fine-tuning.

1 Introduction

Syntactic parse trees are valuable intermediate features for many NLP pipelines (He et al., 2018; Strubell et al., 2018), as a soft constraint (Rush and Collins, 2012), a hard constraint (Lee et al., 2019b), or in multi-task learning with syntactic scaffolds (Swayamdipta et al., 2018). Syntactic inductive bias can also improve generalization of deep learning models (Kuncoro et al., 2020). These results have motivated researchers to pursue unsupervised parsing, with the hope of training syntax-dependent models on large amounts of data without annotation (Klein and Manning, 2002; Bod, 2006; Ponvert et al., 2011; Shen et al., 2019; Kim et al., 2019, inter alia).

Of these models, we focus on the deep inside-outside recursive autoencoder (DIORA; Drozdov et al. 2019a). DIORA encodes sentences in a procedure resembling the inside-outside algorithm (Baker, 1979), which allows it to induce syntactic tree structures for input sentences *without access to labeled training data*, and achieves near state-of-the-art results on *unsupervised* constituency parsing. DIORA resembles pre-trained language models, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), in that it is trained with a self-supervised blank-filling objective on large amounts of unlabeled data.

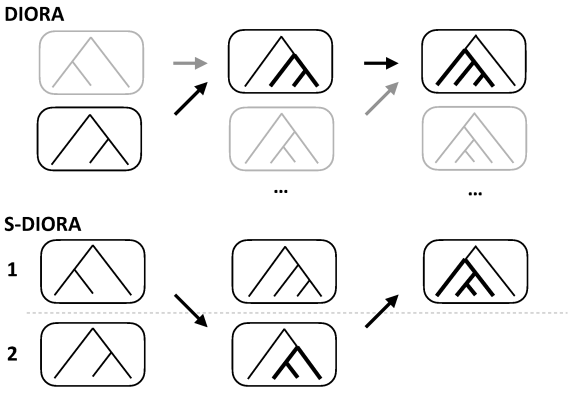

Figure 1: DIORA (top row) is sensitive to locally non-optimal decisions. By assigning a low weight to a potentially important subtree when recursively computing the vector for a target tree, it is difficult or impossible to recover and the important subtree is *washed out* (represented in light gray). Our method, S-DIORA (bottom row) can recover from errors, and the desired tree ends up at the top of the beam in the right-most column.

DIORA is a strong unsupervised parser *in spite of* its locally greedy nature. DIORA works by encoding all subtrees covering a particular span as separate vectors, and then computing a weighted average of these vectors — DIORA uses this averaged vector later in the dynamic program to represent the entire forest of trees covering a span. DIORA computes a score for each subtree; intuitively, a subtree's score affects how strongly it is

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4832–4845,
November 16–20, 2020.

represented in the averaged vector. The representations are computed recursively, and when a tree that looks locally not important is given a weak score, as shown in Figure 1, it will be *washed out*. This weakness in local decision making is similar to the *label bias problem* (Lafferty et al., 2001) in sequence prediction.

In this paper, we extend DIORA so that it can easily recover from local errors (§3). We replace the weight assignment used for vector averaging with a sparse operator equivalent to a one-hot argmax function, ensuring that each representation accurately encodes a **single tree** (hence, we call our method S-DIORA). In S-DIORA, it is not possible for a subtree to be *washed out*, although it is still possible to make an error by ignoring a potentially important subtree. Fortunately, this can be alleviated by adding a beam to each cell of the chart, allowing multiple subtrees over any span to be considered. The key benefit of our modification is that error recovery is easily possible, where previously the vector serves as a bottleneck that makes error recovery difficult or impossible.

We initialize an instance of S-DIORA using the previously released DIORA model, then fine-tune before evaluating on the target domain, constituency parse trees from the English WSJ Treebank (PTB, Marcus et al. 1993). In one experimental setting, we assume no access to the evaluation domain and use a subset of DIORA's training data, a concatenation of the SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2018b) corpora (hereinafter NLI). In the other setting, we assume access to raw text in the target domain, parse tree labels excluded. In both cases, we see S-DIORA improves on the original DIORA performance by at least **4** F1, and training on the PTB raw text leads to more than **3** F1 over the previous state of the art in constituency parsing.

In summary, the main contributions in this paper are: (a) An extension to DIORA called S-DIORA that allows for easy recovery from local errors; (b) New results in unsupervised constituency parsing, improving over the previous state of the art by $\mathbf{2.2-6}\%$ F1 depending on the data used for fine-tuning; and (c) Thorough error analysis of the parse tree output revealing useful insights of why S-DIORA improves over baselines, for example, capturing marginally less prepositional phrases in the parse tree output yet making half the PP-attachment errors.

2 DIORA (Drozdov et al., 2019a)

Drozdov et al. (2019a) introduced DIORA, an unsupervised model that learns to 'reconstruct the input by discovering and exploiting syntactic regularities of the text.' It operates much like a masked language model or denoising autoencoder — first it encodes all-but-one of the words from the input sentence as a vector representation, then it decodes from this vector the missing word. DIORA encodes the sentence in the shape of a constituency tree, yet the model is trained using raw text only and without access to tree annotations. The 'ground truth' tree is unknown, so all valid trees are considered simultaneously using an efficient dynamic program with soft vector weighting.

Here is a sketch for how this approach works. Consider the hypothetical sentence with tokens: $x_0x_1x_2x_3$. Although the 'ground truth' tree is unknown, one valid tree is $\tau = ((x_0(x_1x_2))x_3)$. For each span of token $x_{i:j}$ DIORA computes an *inside* vector $\boldsymbol{h}^{in}_{i,j}$, summarizing the information in that span. Additionally, DIORA computes an *outside* vector $\boldsymbol{h}^{out}_{i,j}$ representing the tokens not in $x_{i:j}$. Assume that x_2 is the target token to predict, then for the parse tree τ the token x_1 is in the *inside* context for x_2 because x_1 is the immediate sibling of x_2 in the subtree capturing both tokens. The tokens x_0 and x_3 are not captured in this subtree and are considered to be in the *outside* context of x_2. DIORA represents the *inside* context as $\boldsymbol{h}^{in}_{1,1}$ and the *outside* context as $\boldsymbol{h}^{out}_{1,2}$ to compute $\boldsymbol{h}^{out}_{2,2,k}$. The k in the subscript indicates that this is only one of many possible valid trees for the hypothetical sentence. DIORA assigns a weight to each valid tree $s_{2,2,k}$ where higher weight values indicate the tree is more helpful for predicting the target token. The vector used to predict the target token is a weighted summation of all the tree representations $\boldsymbol{h}^{out}_{2,2} = \sum_k q_{2,2,k}\boldsymbol{h}^{out}_{2,2,k}$ where $q_{i,j,k}$ is a weight DIORA assigns to each subtree.

The rest of this section covers in more technical detail how to recursively compute the inside and outside vectors and weights for DIORA. The recursive computation is done efficiently using a chart data structure and dynamic program similar to the inside-outside algorithm (Baker, 1979). Part of this computation involves a softly weighted summation, which is an efficient way to encode all valid trees, yet has some downsides (§2.3).

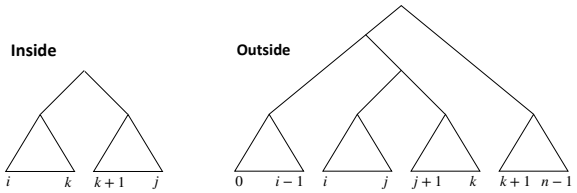

Figure 2: In the inside pass (left) DIORA composes two neighboring vectors. In the outside pass (right) DIORA computes the values for a target span (i, j) recursively from its sibling *inside* span $(j+1, k)$ and outside spans $(0, i-1)$ and $(k+1, n-1)$. The sibling span on the outside pass can appear to the left of the target span, in which case the indexing is adjusted.

2.1 Scoring and Composition

To fill the chart, DIORA learns to compose vectors using a multi-layer neural network (referred to as MLP), and to score vectors using a bi-linear function. In this section, we describe the chart-filling procedure from Drozdov et al. (2019a) using the indexing scheme as demonstrated by Figure 2. The exact equations used to fill the inside chart are:

$$\boldsymbol{h}^{in}_{i,j,k} = \text{MLP}^{in}(\boldsymbol{h}^{in}_{i,k}; \boldsymbol{h}^{in}_{k+1,j})$$
$$s^{in}_{i,j,k} = (\boldsymbol{h}^{in}_{i,k})^\top W \boldsymbol{h}^{in}_{k+1,j} + s^{in}_{i,k} + s^{in}_{k+1,j}$$

In the inside chart, when $i = j$ the scalars s^{in} equal 0, the matrix W is learned, and the vectors $\boldsymbol{h}^{in}$ are equal to the embedding of the token for the i-th position in the sentence x.

The equations for filling the outside chart are:

$$\boldsymbol{h}^{out}_{i,j,k} = \begin{cases} \text{MLP}^{out}(\boldsymbol{h}^{in}_{j+1,k}; \boldsymbol{h}^{out}_{i,k}), & \text{if } k > j \\ \text{MLP}^{out}(\boldsymbol{h}^{in}_{k,i-1}; \boldsymbol{h}^{out}_{k,j}), & \text{else} \end{cases}$$

$$s^{out}_{i,j,k} = \begin{cases} (\boldsymbol{h}^{in}_{j+1,k})^\top U \boldsymbol{h}^{out}_{i,k} + s^{in}_{j+1,k}, & \text{if } k > j \\ \qquad + s^{out}_{i,k} \\ (\boldsymbol{h}^{in}_{k,i-1})^\top U \boldsymbol{h}^{out}_{k,j} + s^{in}_{k,i-1}, & \text{else} \\ \qquad + s^{out}_{k,j} \end{cases}$$

In the outside chart, when $i = 0$ and $j = |S| - 1$ the scalars s^{out} equal 0, the matrix U is learned, and the vector $\boldsymbol{h}^{out}$ is learned independent of the sentence (analogous to the initial hidden state in a recurrent neural network).

For a given span (i, j) there may be multiple valid split points or parent-sibling contexts. If each was considered separately, this would lead to a combinatorial explosion of paths to explore. Instead, DIORA averages the scalars and vectors that share the same (i, j) values. This is identical for the outside or inside pass, taking the following form:

$$q_{i,j,k} = \frac{s_{i,j,k}}{\sum_{k'} s_{i,j,k'}}$$
$$\boldsymbol{h}_{i,j} = \sum_k q_{i,j,k} \boldsymbol{h}_{i,j,k}$$
$$s_{i,j} = \sum_k q_{i,j,k} s_{i,j,k}$$

2.2 Learning

DIORA is trained end to end via word prediction. The bottom-most vectors in the outside chart represent the entire sentence x except for a single token. By predicting this missing token x_i from the outside vector $\boldsymbol{h}^{out}_{i,i}$, we may update the model's parameters without any parse tree labels.[1] The training objective for a single sentence is:

$$J_{rec} = -\frac{1}{|x|} \sum_{i \in |x|} \log P(x_i | \{x\}_{-i}) \qquad (1)$$

2.3 Parse Tree Inference

Although DIORA is not trained with any parse tree annotations, its chart filling procedure can be used to extract binary unlabeled parse trees. First, fill the inside chart following §2.1. Afterwards, use the CKY algorithm to find $\hat{y}$ the maximal scoring tree where the score for a tree y is $S(y) = \sum_{(i,j,k) \in y} s^{in}_{i,j,k}$. This approach demonstrated impressive results for unsupervised constituency parsing (Drozdov et al., 2019a).

To understand better the effectiveness decoding parse trees with DIORA, we train DIORA for *supervised* parsing using a binarized version of the 'ground truth' parse trees from the English Penn Treebank (Marcus et al., 1993). The training procedure is done by optimizing the structured SVM loss:

$$J^{sup}_{tree} = \max(0, S(\hat{y}) - S(y) + 1),$$

where $S(\hat{y})$ is the score of the maximal tree and $S(y)$ is the score of the 'ground truth'.

[1] Since the outside vector is used for word prediction, tricks associated with the inside-outside algorithm using only backpropagation of the inside-pass (Eisner, 2016) are not obviously applicable, if at all.

We use the off-the-shelf parser from Kitaev and Klein (2018) as a baseline and the results are shown in Table 1. Although DIORA is strong in unsupervised parsing, the supervised parsing results are not as competitive with the baseline as we had expected, and lead us to consider deeply why this might be the case.

We posit the low performance in supervised parsing is due to DIORA's inability to effectively recover from local errors. Predicting trees in DIORA is exact — you are guaranteed to find the highest scoring tree given the scalar values associated with each span, but there is a weakness when assigning the scalar values. Specifically, the scalar values are assigned using local information, and may assign a low weight to a subtree which, when given more information, deserves to be given higher weight. Said plainly, this might occur when the sentence has *structural ambiguity* that requires context to resolve. For instance, the clause *'We saw the dog with my phone,'* has a more likely parse tree depending on the context.[2,3] In the next section we present our extension to DIORA that addresses this downside.

	$n \leq 20$		$n \leq 40$	
Model	Binary	N-ary	Binary	N-ary
Kitaev and Klein (2018)	87.5	84.0	85.9	83.6
DIORA	86.0	73.9	81.7	69.1
S-DIORA	89.9	77.5	84.8	73.2

Table 1: Supervised parsing results on the validation set of PTB using parsing F1 with binarized trees. DIORA does not do well because of its inherent weakness, and the best setting from S-DIORA (Table 2) is superior.

3 S-DIORA: Single Tree Encoding

We improve DIORA by making it more robust to local errors. DIORA is sensitive to errors because its vector averaging approach makes it difficult or impossible to recover when important subtrees have been *washed out*. The first modification we present prevents trees from being *washed out* by replacing the weights q with a sparse operator q' equivalent to a one-hot argmax. This effectively replaces vector averaging with selection of the highest scoring subtree for each span.

$$q'_{i,j,k} = \arg\max_{k'}[s_{i,j,k'}][k]$$
$$\boldsymbol{h}_{i,j} = \sum_k q'_{i,j,k} \boldsymbol{h}_{i,j,k}$$
$$s_{i,j} = \sum_k q'_{i,j,k} s_{i,j,k}$$

This change alone is not sufficient. If using only a single highest-scoring tree, S-DIORA would remain as vulnerable, or more so, to local errors that are inevitable when using the context-free approach of the inside-outside algorithm. Instead, at each cell in the chart we record up to β values corresponding to the highest scoring subtrees. We refer to β as beam-size, and our experiments demonstrate that using a beam-size of 2 already gives a great improvement in results, although any size of β can be used at test time regardless what was used during training.

In the popular K-means algorithm, each point minimizes its distance to only one centroid. Using this as motivation, we train S-DIORA s.t. each sentence is only drawn towards one tree. We implement this change using a variant of the structured SVM loss:

$$J^{unsup}_{tree}(x) = \min(0, S(y_1) - S(y_0) + 1),$$

where $S(y_i)$ is the score for the i-th tree represented on the beam and $S(y) = \sum_{(i,j,k) \in y} s^{in}_{i,j,k}$. S-DIORA trains with this loss in addition to the reconstruction loss (the original DIORA objective):

$$J_{\text{S-DIORA}} = J_{rec} + J^{unsup}_{tree}$$

A natural question to ask is whether S-DIORA is difficult to train given $\arg\max$ is relatively non-smooth. To help train S-DIORA, we employed different tricks. During our unsupervised parsing experiments, we used gumbel-top-k (Kool et al., 2019) for q' to ensure the model would sufficiently explore multiple parse trees. We also added regularization via mixout (Lee et al., 2019a) or L2 regularization for the initial parameters so that the model would not diverge drastically from its initialization and suffer catastrophic forgetting. Empirically, we found that none of these additions were necessary, and that fine-tuning with the $J_{\text{S-DIORA}}$ objective was sufficient. One possible explanation for why this is so is that training with $\beta > 1$ already lets S-DIORA explore multiple subtrees for each span during training.

[2] In this particular example, we assume the rest of the sentence serves as sufficient context rather than unavailable information (i.e. world knowledge).

[3] Two valid parses for this clause are: *'(We (saw the dog) (with my phone))'* and *'(We (saw (the dog (with my phone))))'*.

4 Experiments and Results

The model and approach in this paper are motivated mainly by *wash out* in DIORA with respect to its vector averaging. In this section we experimentally test the following hypotheses:

- There are often multiple valid parse trees for a clause (in other words, phrases can have structural ambiguity), therefore we expect that S-DIORA with a beam should be more effective at supervised parsing than DIORA.
- Word prediction benefits from parsing sentences with their most likely constituency tree, therefore S-DIORA, which is trained via word prediction, should be more effective at unsupervised parsing than DIORA because it can recover from local errors.
- Parsers are sensitive to their training domain. Although we expect training with S-DIORA to be helpful for unsupervised parsing, we expect an even bigger benefit when training on the same domain as used for evaluation.

4.1 Preliminaries: Constituency Parsing

We measure the performance of our changes via unsupervised and supervised parsing on the test set of the WSJ Penn Treebank (Marcus et al., 1993).[4] All models (S-DIORA and baselines) output unlabeled binary trees[5] and are evaluated via sentence level F1 (S-F1).

- **True Positives (TP)** are the spans in both parse trees (inferred and ground truth).
- **False Positives (FP)** are spans predicted but not in the ground truth.
- **False Negatives (FN)** are spans in the ground truth but not the predicted tree.
- **Sentence F1** $= \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \frac{2 \times \mathbf{TP}(x)}{2 \times \mathbf{TP}(x) + \mathbf{FP}(x) + \mathbf{FN}(x)}$

Following previous work, we consider only non-trivial spans (covering 2 or more words, and ignoring spans covering the entire sentence). For pre-processing we remove punctuation.

[4] We evaluate all models using the eval script from Kim et al. (2019). We noticed a small bug in the eval script where some spans covering the entire sentence were not being ignored. This lead to a very small change in the numbers, but for this reason, our numbers for baselines may appear slightly different from previous work.

[5] Although models output binary trees, the ground truth has n-ary trees. This establishes an upper bound on the highest possible F1 since each model has an unavoidable penalty to precision.

4.2 Supervised Parsing

For supervised constituency parsing we use the off-the-shelf parser from Kitaev and Klein (2018) as a baseline to compare against DIORA and S-DIORA. For training we use the parse trees from training split of PTB and evaluate using the validation data. We binarize the ground truth using the Stanford parser (Manning et al., 2014) and train for 10 epochs. Results against the binary trees and original n-ary (ingoring labels in both cases) is shown in Table 1. Both DIORA and S-DIORA are trained from random initialization using the structured SVM loss from Kitaev and Klein (2018).

We see that DIORA is not competitive with the Kitaev and Klein (2018) parser, and attribute this to *wash out* and its inability to recover from errors.

For S-DIORA we train and evaluate with $\beta \in \{1, 2, 3, 4\}$ and results are shown in Table 2. We see, unsurprisingly, that regardless of the beam-size at training, when $\beta = 1$ at test time the performance is worse than DIORA. This is because even though S-DIORA does not suffer from *wash out*, when the beam is too small it can not recover from errors. As the beam-size increases, so does performance, surpassing DIORA by 3 F1 in the best case ($\beta = 3$ for training; $\beta = 4$ at test time).

	$n \leq 20$				$n \leq 40$			
β	1	2	3	4	1	2	3	4
1	84.6	87.8	88	88.2	77.7	81.5	82.3	82.6
2	85.1	88.8	89.4	89.7	78.6	83	83.9	84.4
3	85.7	88.9	89.7	**89.9**	79.4	83.3	84.5	**84.8**
4	84.7	88.7	89.3	89.5	78.4	82.8	83.8	84.2

Table 2: Supervised parsing results on the validation set of PTB using binarized trees for S-DIORA. The grid represents parsing F1 with different values of β at train time (rows) and test time (columns). The model is not effective when β at test time is 1 because it can not recover from errors. Increasing β for both training and test test time is helpful, with the best performance for $\beta_{train} = 3$ and $\beta_{test} = 4$. Beam search benefits short sentences (length $n \leq 20$) and long ones ($n \leq 40$).

4.3 Unsupervised Parsing

We explored two settings in unsupervised parsing. In the first, the zero-shot case, we assume no access to the evaluation domain. Instead, we sample a subset from the NLI data used to train DIORA and use this to fine-tune S-DIORA. The subset includes the same number of sentences as the train-

ing data from PTB and the same sentence length distribution. The results are shown in Table 3 with the model name S-DIORA$_{NLI}$. This model does substantially better than the original DIORA (and improvement of more than 5 F1) and is even competitive with the state-of-the-art model C-PCFG (Kim et al., 2019).

In the other experimental setting we assume access to raw text in the target domain is available but annotations are not. We fine-tune using the training data from PTB (about 40k sentences) and results are shown in Table 3 with the model name S-DIORA$_{PTB}$. This improves upon S-DIORA$_{PTB}$ by a full 2 F1 points and is also substantially better than the previous state of the art by 3.5 F1.

S-DIORA sees a large improvement in WSJ-10. These sentences are length 10 or less and previously DIORA was on-par with ON-LSTM (Shen et al., 2019). When we bucket F1 by sentence length, we see that S-DIORA improves not only short sentences but on all sentence lengths.

To determine whether fine-tuning is necessary, we initialize S-DIORA from DIORA and evaluate it immediately. In this setting DIORA$_{None}$ performs 5 F1 less than DIORA, confirming the importance of fine-tuning.

To further determine if the extra training data was the main factor in the improved performance, we train DIORA with the an equivalent amount of data and see no improvement. This is not surprising given that the pre-trained DIORA was trained initially until convergence on relatively more data.

	WSJ		
Model	**F1**$_{max}$	**F1**$_{\mu}$	**F1**$_{n\leq 10}$
ON-LSTM (Shen et al., 2019)	50.21	48.1†	61.02
C-PCFG (Kim et al., 2019)	60.32	55.2†	68.82
DIORA (Drozdov et al., 2019a)	56.75	-	60.55
S-DIORA$_{None}$ (Ours)	51.56	-	59.36
S-DIORA$_{NLI}$ (Ours)	61.68	54.8	70.41
S-DIORA$_{PTB}$ (Ours)	**63.96**	57.6	**71.80**

Table 3: Unsupervised parsing results. We evaluate each model on the full PTB test set using the evaluation script provided by Kim et al. (2019). The average across random seeds is F1$_{\mu}$[6]and the best model's F1 is reported as F1$_{max}$. We take the best model and also evaluate it on sentences of length of 10 or less and report the value in F1$_{n\leq 10}$. Values with a † are copied from Kim et al. (2019). We only had access to a single DIORA model so no F1$_{\mu}$ is reported.

4.4 Training and Implementation Details

When applicable, we use the MLP with 'softmax loss' model checkpoint provided by Drozdov et al. (2019a). S-DIORA makes an impactful change to DIORA, but its parameters are exactly the same, making it easy to load a pretrained DIORA model for S-DIORA. Our implementation of S-DIORA, checkpoints of best models, training scripts, and all parsing output are available online.[7] Additional training details are covered in the Appendix A.1.

5 Discussion and Analysis

In this section we examine the parse tree output of the models in our experimental setup with more fine-grained detail than parsing F1. Given the prevalence of pre-trained language models in NLP tasks, we also include in our analysis recent results using transformers for unsupervised parsing. In addition, we present a new baseline demonstrating that pretrained language models are better at unsupervised parsing than previously known.

5.1 Linguistic Error Analysis

Parsing F1 is useful to quickly compare performance between parsers, and previous work in unsupervised parsing often also report segment recall to give a sense of which phrases are most often captured in the output. To provide an even more thorough treatment of linguistic errors we add labels to the parse trees using the parser from Kitaev and Klein (2018) and then run the Berkeley parser analyzer (Kummerfeld et al., 2012). This latter tool classifies mistakes for each predicted tree by the type of phrases (or patterns like coordination) involved in the error, allowing analysis of the types of errors being made by a model. In Table 4 we show the parsing F1, segment recall, and error counts as determined by the analyzer.

By segment recall, we see that C-PCFG outperforms DIORA in segment recall for NP and PP, explaining its high S-F1. The linguistic analysis tells a slightly different story — *C-PCFG makes less errors associated with NP internal structure and clause attachment, but substantially more errors associated with PP attachment.*

[7] https://github.com/iesl/s-diora

[7] S-DIORA F1$_{\mu}$ is reported across 5 random seeds with the same hyperparameters. S-DIORA$_{None}$ is evaluated after initialization, so F1$_{\mu}$ = F1$_{max}$. Since we use early stopping it is not possible for the best model to be worse than initialization, hence S-DIORA performance is strictly $\geq$ performance of S-DIORA$_{None}$.

Model	β	S-F1	SBAR	NP	VP	PP	ADJP	ADVP	Mod.	NP-I	NP-A	PP-A	VP-A	Clause	Coord.
ELMo	-	42.3	40.3%	50.7%	43.7%	45.9%	57.0%	74.0%	826	1416	211	1943	21	1239	124
XLNet$_{\lambda=0}$	-	40.8	35.3%	57.1%	28.1%	37.4%	53.0%	58.0%	974	1348	221	1935	**14**	1423	**103**
XLNet$_{\lambda=1.5}$	-	48.0	60.9%	52.8%	51.7%	56.2%	51.4%	68.3%	673	1347	**182**	1748	50	1375	117
DIORA	-	56.9	68.1%	74.2%	61.4%	55.1%	54.7%	74.4%	634	784	237	1356	47	928	165
C-PCFG	-	61.2	62.1%	82.0%	53.5%	69.7%	54.0%	62.6%	655	**753**	253	1997	42	858	166
S-DIORA$_{\text{None}}$	1	50.6	55.5%	67.4%	48.2%	48.0%	54.7%	64.1%	837	931	281	1629	44	939	174
S-DIORA$_{\text{NLI}}$	1	59.7	55.4%	73.5%	72.9%	59.9%	46.0%	53.1%	642	952	293	1076	76	**704**	180
S-DIORA$_{\text{PTB}}$	1	61.9	56.8%	76.3%	76.4%	65.9%	43.9%	60.7%	537	922	281	930	84	933	187
S-DIORA$_{\text{None}}$	3	51.6	59.4%	67.4%	51.5%	48.4%	57.5%	61.1%	803	945	252	1432	46	910	164
S-DIORA$_{\text{NLI}}$	3	61.3	58.0%	75.2%	76.5%	61.2%	50.9%	56.9%	585	920	292	910	76	753	177
S-DIORA$_{\text{PTB}}$	3	63.3	59.2%	78.0%	78.9%	67.1%	49.1%	59.9%	**487**	917	265	**861**	91	954	186

Table 4: To better understand the difference between models, displayed above are the segment recall on the WSJ validation set separated by phrase type (the left columns). For a more informative look at linguistic phenomenon, we use the Berkeley parser analyzer (Kummerfeld et al., 2012) and display error counts (the right columns). Since the unsupervised parsing models do not provide labels, we use high performing supervised constituency parser (Kitaev and Klein, 2018) to label the trees. β is beam size. The frequency of each label in the validation set is ADJP=428, ADVP=262, NP=10350, PP=3877, SBAR=1091, VP=5407. The error types are: Modifier Attachment (Mod.), NP Internal Structure (NP-I), NP Attachment (NP-A), PP Attachment (PP-A), Clause Attachment (Clause), and Coordination (Coord).

5.2 Unsupervised Parsing with Large Pre-trained LMs

We introduce a new unsupervised parsing baseline using ELMo (Peters et al., 2018), so that we may compare S-DIORA with large pre-trained LMs, a class of models that have recently proven very effective across NLP tasks. To extract a parse tree from ELMo, we first compute vector similarity between phrase embeddings in the output, then use these scalar values as input to the CKY algorithm.[8]

Compared to ELMo we see that S-DIORA captures less ADVP phrases yet also makes less NP-I errors. Although S-DIORA has a strong affinity for VP phrases ELMo makes less VP-A errors.

For further comparison we include the best models from Kim et al. (2020). We see that XLNet$_{\lambda=0}$ is the worst of all models in S-F1 and VP segment recall, but also has the fewest VP-A errors. This suggests that errors related to segment recall are likely folded into a different category such as PP attachment. The right-skewed model XLNet$_{\lambda=1.5}$ substantially improves over XLNet$_{\lambda=0}$ in SBAR recall and is comparable in this category with S-DIORA.

Interestingly, although increasing the size of β in S-DIORA results in a *near monotonic improvement in all categories* (with some minor exceptions), S-DIORA shows a very different error profile when compared to pre-trained LMs, despite having a better S-F1. For instance, the pre-trained LMs make fewer coordinations errors, and perform better with adverbial phrases (ADVP), than any version of S-DIORA. In future work, it may be useful to understand why parser performance does not increase monotonically. Perhaps this is an artifact of the current state of unsupervised parsing research and will change once parsers improve beyond some threshold.

5.3 The Benefit of Error Recovery

5.3.1 DIORA versus S-DIORA

It is not sufficient to initialize S-DIORA from DIORA without fine-tuning. DIORA$_{None}$ does worse than DIORA in nearly every category. Furthermore, the biggest benefit is gained when using S-DIORA with $\beta > 1$, otherwise error recovery is not possible (see Figure 3).

DIORA is trained on NLI and it is not surprising it incurs so many errors in coordination and clause attachment, which are frequently observed in domain mis-match (Kummerfeld et al., 2012). We used the same checkpoint for finetuning with the original formulation of DIORA — any improvements would be from exposure to more training data. When using NLI for finetuning, across 5

[8] To compute phrase embeddings, we follow the procedure from (Kitaev and Klein, 2018) which concatenates the forward and backward LSTM vectors at the beginning and end of each phrase. To compute vector similarity we follow the procedure in Kobayashi et al. (2019) which uses ELMo sentence embeddings for RST parsing — rather than document level parsing, our work pertains to sentence level parsing.

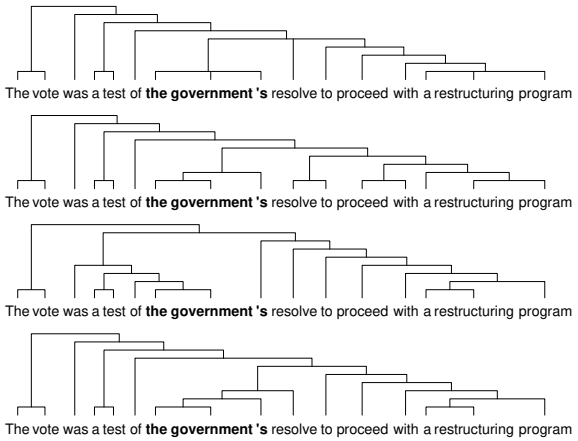

Figure 3: In this example, a beam-size of 1 is not sufficient for S-DIORA to improve upon DIORA — error recovery is only achieved with larger β. The trees from top to bottom are from PTB, DIORA, then S-DIORA$_{PTB}$ with $\beta = 1, 3$. Although larger β can lead to more errors in certain situations (specifically clausal attachment), here they decrease.

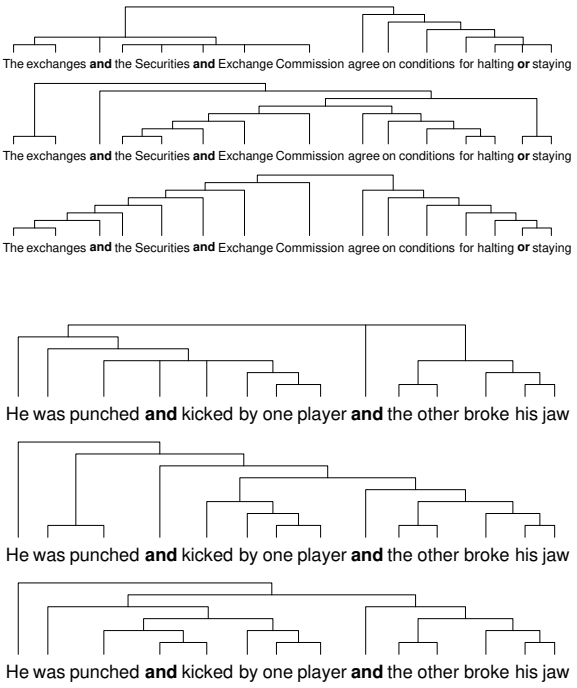

Figure 4: Two sentences where beam-search helps with ambiguous coordination structures, correctly nesting noun phrases (top) and getting better coordination of verb phrases (bottom). The displayed parse tree output, top to bottom, are from PTB, then S-DIORA$_{PTB}$ with $\beta = 1, 3$ respectively.

random seeds there was no improvement over the pre-trained model. This is not surprising given the original models were trained until convergence with relatively large amounts of training data.

Training on NLI provides S-DIORA with a substantial advantage in segment recall for VP and PP. S-DIORA does much worse in capturing the low frequency ADVP category. This does not incur much penalty in S-F1 but is reflected in NP-I.[9]

5.3.2 Effects of Beam Size

Performance improves across the board as we increase beam size β, and S-DIORA$_{PTB}$ improves over DIORA suggesting that single tree encoding already provided some benefit (recall that we fine-tuned DIORA on both NLI and PTB with no improvements in unsupervised parsing). Most benefit is achieved using $\beta = 3$, although in some cases it helps to increase it further (see Figure 5). Increasing the beam also helps with different classes of errors. In Figure 4 we see the benefit in sentences with tricky coordination.

5.4 Labeled Parsing

We evaluate the labeled trees from §5.1, and the best performing S-DIORA model achieves 80.7 *labeled* parsing F1 on the validation data (72.3 recall, 91.2 precision, and 11.7 complete match) when evaluated this way. *This suggests that unsupervised parsers are closer to supervised parsers than previously realized*, and although deciding which phrases are in the tree is the harder task (Klein and Manning, 2002), it may be worth pursuing unsupervised labeling[10] for more informative error analysis (Bisk and Hockenmaier, 2015).

6 Related Work

Avoiding errors by using rich feature models. The nature of *unsupervised* parsing is that good performance is a result of strong inductive bias, explaining why DIORA and S-DIORA are so effective, yet their context-free approach to chart parsing is also the cause of local errors. S-DIORA employees a beam at each cell to recover from local errors, but this would be less helpful if errors were less frequent. Top performing *super-*

[9]The NP-I category covers missed gold phrases within large noun phrases. In general, much of NP structure in PTB is not annotated, and in future work it is worth using the data provided by Vadas and Curran (2011) to investigate NP structure, as determined by unsupervised parsers, more thoroughly.

[10]Typically unsupervised constituency parsing is purely evaluated by its structure, although recent work from Drozdov et al. (2019b) shows that a simple approach to induce labels with DIORA can be done by clustering the inside and outside phrase vectors.

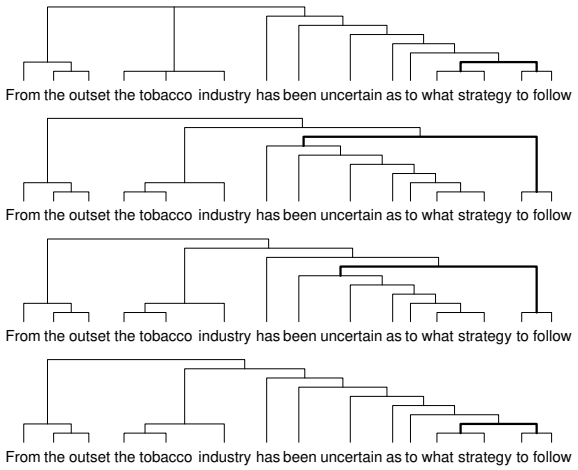

Figure 5: As the beam-size increases, S-DIORA's output tends to match the ground truth more closely. The displayed output, top to bottom, are from PTB, then S-DIORA$_{PTB}$ with $\beta = 1, 3, 5$ respectively.

vised parsers do not need error recovery because they use models with rich features and model each span score independently (Cross and Huang, 2016; Stern et al., 2017; Kitaev and Klein, 2018; Mrini et al., 2019). Previous research has attempted to achieve the "best of both worlds" by distilling a strong model for *supervised* parsing via an *unsupervised* model's output (Le and Zuidema, 2015).

These approaches are closely related to **fast and accurate parsing**. More accurate models tend to use richer features that are more expensive to compute, influencing researchers to find efficient techniques to offset the loss in speed (Vieira and Eisner, 2017). In this paper, we use the most simple approach to learn to parse with the capability to recover from local errors by maintaining a beam of size β at each cell in the chart. S-DIORA is often faster and discovers better trees than DIORA, but there are other methods for extracting lists of best or plausible parses (Resnik, 1992; Roark and Johnson, 1999; Charniak and Johnson, 2005; Huang and Chiang, 2005; Bouchard-côté et al., 2009) that might further improve performance.

Sparse structured inference. Various work has explored sparse alternatives to soft-weighting. Sparsemax (Martins and Astudillo, 2016) is a deterministic sparse alternative to the softmax, and Gumbel-Softmax (Jang et al., 2017) uses the categorical reparameterization trick to sample a discrete value during training. Both have attractive properties but alone would not be sufficient for overcoming local errors in S-DIORA. Nonetheless, these options would be worth exploring for unsupervised parsing when training with more data or when the ground truth parse trees are very different than the ones in S-DIORA's output frontier after initialization. Other work has explored methods for differentiable structured inference (Niculae et al., 2018; Mensch and Blondel, 2018; Corro and Titov, 2019a,b), which may also be suitable. It's worth noting that PCFGs are not graphical models (Liang et al., 2009), and marginal inference is often not tractable,[11] which is why these approximate methods may be helpful.

Grammar induction. There is a rich research history in grammar induction and unsupervised parsing (Fu and Booth, 1975; Angluin, 1980; Carroll and Charniak, 1992). We cover notable work not already mentioned in Appendix A.2.

7 Conclusion

We introduce S-DIORA, an extension to DIORA that enables for easy recovery from local errors and is not subject to *wash out* from vector averaging. Our experiments in supervised parsing verify S-DIORA improves upon the representational power of DIORA. Unsupervised fine-tuning with S-DIORA leads to new impressive results in unsupervised constituency parsing, improving upon the previous state of the art by $\mathbf{2.2 - 6}\%$ F1, depending on the data used.

Acknowledgements

We are grateful to our colleagues at UMass for help and advice, and to the UMass NLP reading group and the anonymous reviewers for feedback on drafts of this work. This work was supported in part by the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative, and in part by the National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1514053, CNS-0958392, and IIS-1955567. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

[11] Although marginal inference is effectively leveraged in C-PCFG (Kim et al., 2019), the recursive vector composition in DIORA and S-DIORA makes this intractable. McAllester et al. (2008) propose case-factor diagrams which can efficiently encode PCFGs and compute the normalizing partition function, although it is not clear how to leverage them for self-supervised training with neural networks.

References

Dana Angluin. 1980. Inductive inference of formal languages from positive data. *Information and control*, 45(2):117–135.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Speech communication papers presented at th 97th Meeting of the Acoustical Society of America*, pages 547–550.

Yonatan Bisk and Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Association for Computational Linguistic (ACL)*.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Association for Computational Linguistics (ACL)*.

Alexandre Bouchard-côté, Slav Petrov, and Dan Klein. 2009. Randomized pruning: Efficiently calculating expectations in large dynamic programs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Glenn Carroll and Eugene Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *AAAI Workshop on Statistically-Based NLP Techniques*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Association for Computational Linguistics (ACL)*.

Caio Corro and Ivan Titov. 2019a. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In *International Conference on Learning Representations (ICLR)*.

Caio Corro and Ivan Titov. 2019b. Learning latent trees with stochastic perturbations and differentiable dynamic programming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*.

Andrew Drozdov, Pat Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019a. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. In *North American Association for Computational Linguistics (NAACL)*.

Andrew Drozdov, Patrick Verga, Yi-Pei Chen, Mohit Iyyer, and Andrew McCallum. 2019b. Unsupervised labeled parsing with deep inside-outside recursive autoencoders. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2009. Semi-supervised learning of dependency parsers using generalized expectation criteria. 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*.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*. Association for Computational Linguistics (ACL).

King-Sun Fu and Taylor L Booth. 1975. Grammatical inference: Introduction and survey-part ii. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):409–423.

Kevin Gimpel and Mohit Bansal. 2014. Weakly-supervised learning with cost-augmented contrastive estimation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Association for Computational Linguistics (ACL)*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Vancouver, British Columbia. Association for Computational Linguistics (ACL).

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.

Taeuk Kim, Jihun Choi, Daniel Edmiston, and Sang goo Lee. 2020. Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction. In *International Conference on Learning Representations (ICLR)*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019. Compound probabilistic context-free grammars for grammar induction. In *Association for Computational Linguistics (ACL)*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Association for Computational Linguistic (ACL)*.

Dan Klein and Christopher D Manning. 2002. Natural language grammar induction using a constituent-context model. In *Advances in neural information processing systems (NeurIPS)*.

Naoki Kobayashi, Tsutomu Hirao, Kengo Nakamura, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. 2019. Split or merge: Which is better for unsupervised RST parsing? In *EMNLP*.

Philipp Koehn and Kevin Knight. 2003. Feature-rich statistical translation of noun phrases. In *proceedings of the 41st Annual Meeting of the association for Computational Linguistics*, pages 311–318.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning (ICML)*.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *arXiv preprint arXiv:2005.13482*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019a. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations (ICLR)*.

Jay Yoon Lee, Sanket Vaibhav Mehta, Michael R. Wick, Jean-Baptiste Tristan, and Jaime G. Carbonell. 2019b. Gradient-based inference for networks with output constraints. In *AAAI*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Probabilistic grammars and hierarchical dirichlet processes. In *The Handbook of Applied Bayesian Analysis*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics (ACL): System Demonstrations*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

André F. T. Martins and Ramón Fernández Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*.

David McAllester, Michael Collins, and Fernando Pereira. 2008. Case-factor diagrams for structured probabilistic modeling. *Journal of Computer and System Sciences*.

Arthur Mensch and Mathieu Blondel. 2018. Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning (ICML)*.

Anhad Mohananey, Katharina Kann, and Samuel R. Bowman. 2020. Self-training for unsupervised parsing with prpn. In *IWPT*.

Khalil Mrini, Franck Dernoncourt, Trung Bui, Walter Chang, and Ndapa Nakashole. 2019. Rethinking self-attention: An interpretable self-attentive encoder-decoder parser. *arXiv preprint arXiv:1911.03875*.

Tahira Naseem and Regina Barzilay. 2011. Using semantic cues to learn syntax. In *AAAI*.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Vlad Niculae, André F. T. Martins, Mathieu Blondel, and Claire Cardie. 2018. Sparsemap: Differentiable sparse structured inference. In *ICML*.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *30th Annual Meeting of the Association for Computational Linguistics*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics (NAACL)*.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Association for Computational Linguistics (ACL)*.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *COLING 1992 Volume 1: The 15th International Conference on Computational Linguistics*.

Brian Roark and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 421–428, College Park, Maryland, USA. Association for Computational Linguistics.

Alexander M Rush and MJ Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations (ICLR)*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *Association for Computational Linguistics (ACL)*.

Noah A Smith and Jason Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

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

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *ACL/IJCNLP*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Empirical Methods in Natural Language Processing (EMNLP)*.

David Vadas and James R. Curran. 2011. Parsing noun phrases in the penn treebank. *Computational Linguistics*.

Tim Vieira and Jason Eisner. 2017. Learning to prune: Exploring the frontier of fast and accurate parsing. *Transactions of the Association for Computational Linguistics*, 5:263–278.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2018a. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association of Computational Linguistics (TACL)*, 6:253–267.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Association for Computational Linguistics (NAACL)*.

A Appendices

A.1 Training Details

All key details for training and evaluating our method, S-DIORA, are described in the main text. In this Appendix section we repeat those details and provide an organized reference to aid reproducibility.

A.1.1 Supervised Parsing Loss and Training

In supervised parsing, we assume access to binary non-projective constituency trees y for each sentence x. Predicting a tree $\hat{y}$ with DIORA can be done using the CKY method described in Drozdov et al. (2019a). Similarly, backtracking the various max operations from the inside-pass in S-DIORA can be used to decode $\hat{y}$.[12] The conditional probability of a tree given a sentence is proportional to the sum of scalar values for each span and split (i, j, k) in the tree, depicted in Eq. 2.

$$P(y|x) \propto S(y) = \sum_{(i,j,k) \in y} s^{in}_{i,j,k} \quad (2)$$

To train DIORA or S-DIORA to predict the most likely tree for an input sentence, we use the structured SVM loss employed by multiple other work in supervised parsing (Stern et al., 2017; Kitaev and Klein, 2018) with a margin of 1 and do not use loss augment inference, depicted in Eq. 3.

$$J^{sup}_{tree} = \max(0, S(\hat{y}) - S(y) + 1) \quad (3)$$

In our experiments, we train DIORA and S-DIORA on the training from PTB (roughly 40k sentences). Both models are trained from random initialization and using the same hyperparameters. Early stopping is done by evaluating against the validation data each epoch. S-DIORA is trained with different beam-size $\beta = \{1, 2, 3, 4\}$.

This paper is primarily concerned with *unsupervised* parsing, and we only explored one hyperparameter setting as supervised parsing is used primarily to verify the benefit of beam search in S-DIORA and its improvement over DIORA. Those hyperparameters are listed here:

Learning Rate (η):	2^{-3}
Model Dimension:	400
Max Training Length:	20
Batch Size:	32
Max Epochs:	10
Optimization Algorithm:	Adam
Hardware:	1x1080ti
Training Time:	O(12h)

For both supervised and unsupervised training, each batch is restricted to sentences of uniform length.

A.1.2 Unsupervised Loss and Training

DIORA and S-DIORA are models especially effective for unsupervised parsing. In this setting we assume no access to parse tree labels, only raw text. The models are trained end-to-end by reconstructing the input sentence from the outside vectors. Reconstruction is defined as predicting a word x_i given its context $\{x\}_{-i}$ which are the words in the rest of the sentence. Unlike Drozdov et al. (2019a), we use a fixed vocabulary instead of sampling, which includes the 10k most frequent words from the training data.[13] The objective for a single sentence is depicted in Eq. 4.

$$J_{rec} = -\frac{1}{|x|} \sum_{i \in |x|} \log P(x_i | \{x\}_{-i}) \quad (4)$$

As mentioned in §3, we also train S-DIORA to increase the confidence gap between its highest-scoring tree on the beam and other trees. To accomplish this we use the same structured SVM from supervised parsing, but instead of the ground truth y, we include the highest-scoring tree on the beam y_0 and the second highest y_1. This loss is depicted in Eq. 5, and the total loss for S-DIORA is simply the sum of the reconstruction and 'tree' losses (Eq. 6).

$$J^{unsup}_{tree} = \max(0, S(y_1) - S(y_0) + 1) \quad (5)$$

$$J_{\text{S-DIORA}} = J_{rec} + J^{unsup}_{tree} \quad (6)$$

For S-DIORA$_{NLI}$ we train using a subset of NLI.[14] The subset is sampled once from NLI and

[12] Each value save on the beam in S-DIORA represents a unique tree — duplicate trees can not appear on the beam.

[13] The vocabulary is different between NLI and PTB.

[14] A concatenation of the training data from SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2018b).

used across all experiments, and consists of the same number of sentences as the training data from PTB and also the same distribution of sentence lengths. For S-DIORA$_{PTB}$ we use the training data from PTB. Early stopping is done by evaluating against the validation data each epoch. We explore various hyperparameter settings, and for S-DIORA we also train with different beam-sizes β. S-DIORA is initialized from the MLP with 'softmax loss' DIORA checkpoint[15] that was released by Drozdov et al. (2019a). The hyperparameters explored are listed below:

Learning Rate (η):	$2^{-3}, 1^{-3}, 6^{-4}, 2^{-4}$
Model Dimension:	400
Beam-size (β):	$2, 3$
Max Training Length (n):	$20, 30$
Batch Size:	32
Max Epochs:	5
Optimization Algorithm:	Adam
Hardware:	1x1080ti
Training Time:	O(8h)

We ran each setting for 5 random seeds. The best performing hyperparameter setting was chosen using validation performance, and the best performing setting (η, β, n) for S-DIORA$_{NLI}$ and S-DIORA$_{PTB}$ were $(1^{-3}, 2, 30)$ and $(2^{-3}, 2, 30)$ respectively.

A.2 Other Work in Grammar Induction and Unsupervised Parsing

There is a rich research history in grammar induction and unsupervised parsing. In the main text, we cover the work most relevant to frame our scientific questions and experimental results. Instead, here, we mention loosely related work that would be useful for further analysis and future research. Furthermore, some of the mentioned work might be in dependency parsing rather than constituency parsing, or about measuring syntactic information without parse trees.

[15]DIORA and S-DIORA have exactly the same parameters, so one can be initialized easily from the other. The number of parameters is the same, but the runtime of S-DIORA is slower by an order of β. Even so, a correctly implemented S-DIORA should be as fast as DIORA or faster since the sparse operator q' can be leveraged to avoid computation when there are many possible subtrees for a span.

A.2.1 Partial Semantic Information

We assume access to no text annotation, but often some might be available (Pereira and Schabes, 1992) and this can be leveraged to constrain induced syntax in a useful way. Naseem and Barzilay (2011) explore syntactic structure of semantic relations, presenting an approach that encourages structural consistency for each occurrence of a specific semantic relation, but also allowing for variation. DIORA and S-DIORA represent spans as vectors, and a simple extension would be to encourage span vectors associated with the same semantic relation to be similar through contrastive estimation (Smith and Eisner, 2005a,b; Gimpel and Bansal, 2014). Rather than encouraging similarity within a relation, Shi et al. (2019) have success encouraging similarity between an image and constituents in its caption.

A.2.2 Multilingual Alignment

Syntactic phrase types do not necessarily translate to the same type across languages (Koehn and Knight, 2003), but can still leverage parallel text to improve unsupervised constituency parsing as a phrase in one language may have less uncertain structure in another (Snyder et al., 2009).

A.2.3 Label Refinement

Similarities across languages can be used to create fine-grained grammar rules that are helpful when applied as soft constraints for grammar induction since they serve as a prior to contradict patterns seen in the data (Naseem et al., 2010). These linguistic priors need not be derived from cross-lingual data (Druck et al., 2009) — using a small set of simple rules (e.g. a determiner followed by a noun is a noun phrase) can be helpful for grammar induction and can be derived from a few positive examples of phrases (Haghighi and Klein, 2006).

A.2.4 Model Consistency

Williams et al. (2018a) measure *self F1* in addition to parsing F1 and find the models that consistently converge to the same grammar were also the ones most different from ground truth, although this was an extreme case as the pertinent model made trivial predictions (nearly always left-branching). Follow up work from Mohananey et al. (2020) shows that self-training is helpful for training PRPN (Shen et al., 2018) and parsing F1 improves with self-agreement, with the biggest benefit for longer sentences.