Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Autoencoders

Andrew Drozdov*, **Pat Verga***, **Mohit Yadav***, **Mohit Iyyer**, **and Andrew McCallum**

College of Information and Computer Sciences
University of Massachusetts Amherst

{adrozdov, pat, ymohit, miyyer, mccallum}@cs.umass.edu

Abstract

We introduce deep inside-outside recursive autoencoders (DIORA), a fully-unsupervised method for discovering syntax that simultaneously learns representations for constituents within the induced tree. Our approach predicts each word in an input sentence conditioned on the rest of the sentence and uses inside-outside dynamic programming to consider all possible binary trees over the sentence. At test time the CKY algorithm extracts the highest scoring parse. DIORA achieves a new state-of-the-art F1 in unsupervised binary constituency parsing (unlabeled) in two benchmark datasets, WSJ and MultiNLI.

1 Introduction

Syntactic parse trees are useful for downstream tasks such as relation extraction (Gamallo et al., 2012), semantic role labeling (Sutton and McCallum, 2005; He et al., 2018), machine translation (Aharoni and Goldberg, 2017; Eriguchi et al., 2017; Zaremoodi and Haffari, 2018), and text classification (Li and Roth, 2006; Tai et al., 2015). Traditionally, supervised parsers trained on datasets such as the Penn Treebank (Marcus et al., 1993) are used to obtain syntactic trees. However, the treebanks used to train these supervised parsers are typically small and restricted to the newswire domain. Unfortunately, models trained on newswire treebanks tend to perform considerably worse when applied to new types of data, and creating new domain specific treebanks with syntactic annotations is expensive and time-consuming.

Motivated by the desire to address the limitations of supervised parsing and by the success of large-scale unsupervised modeling such as ELMo and BERT (Peters et al., 2018a; Devlin et al., 2019), we propose a new deep learning method of unsupervised parser training that can extract both shallow parses (i.e., noun phrases or entities) and full syntactic trees from any domain or language automatically *without requiring any labeled training data*. In addition to producing parses, our model simultaneously builds representations for internal constituents that reflect syntactic and semantic regularities which can be leveraged by downstream tasks.

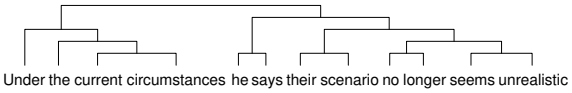

Figure 1: An unlabeled binary constituency parse from DIORA matching the ground truth.

Our model builds on existing work developing latent tree chart parsers (Socher et al., 2011b; Le and Zuidema, 2015; Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2018). These methods produce representations for all internal nodes in the tree (cells in the chart), each generated as a soft weighting over all possible sub-trees (§2). Unfortunately, they still require sentence-level annotations during training, as they are all trained to optimize a downstream task, typically natural language inference.

To address these limitations, we present deep inside-outside recursive autoencoders (DIORA) which enable unsupervised discovery and representation of constituents without requiring any supervised training data. DIORA incorporates the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) into a latent tree chart parser. The bottom-up inside step calculates a representation for all possible constituents within a binary tree over the input sentence. This step is equivalent to the forward-pass of previous latent tree chart parsers (Maillard et al., 2017). These inside representations only encode the current subtree, ignor-

*Equal contribution, randomly ordered.

Proceedings of NAACL-HLT 2019, pages 1129–1141
Minneapolis, Minnesota, June 2 - June 7, 2019.

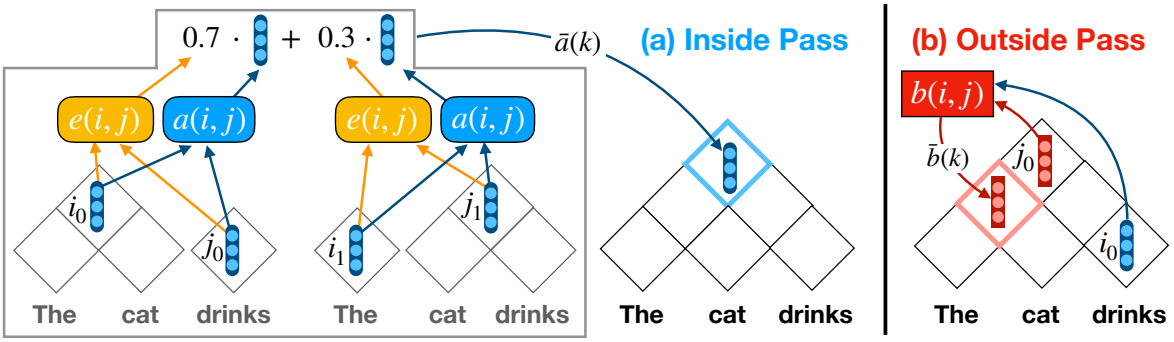

Figure 2: The illustrated inside and outside pass of DIORA operating over an input of length three, 'the cat drinks'. a) The inside pass: The inside vector $\bar{a}(k)$ for the phrase 'the cat drinks' is a weighted average of the compositions for the two possible segmentations - ((the cat), drinks) and (the, (cat drinks)). The scalar weights come from a learned compatibility function. b) The outside pass: The outside vector $\bar{b}(k)$ for the phrase 'the cat' is a function of the outside vector of its parent 'the cat drinks' and the inside vector of its sibling 'drinks'.

ing all outside context. Thus, we perform an additional top-down outside calculation for each node in the tree, providing external context into the subtree representations in each chart cell. The model is then trained with the objective that the outside representations of the leaf cells should reconstruct the corresponding leaf input word, analogous to masked language model (Devlin et al., 2019) pre-training, except by using dynamic programming we predict every word from a completely unmasked context. The single most likely tree can be recovered using the CKY algorithm and compatibility scores between constituents. Previous work either predict trees that are not well aligned with known treebanks (Yogatama et al., 2017; Choi et al., 2018), or has no mechanism for explicitly modeling phrases, requiring a complex procedure to extract syntactic structures (Shen et al., 2018).

To probe different properties of our model, we run experiments on unsupervised parsing, segment recall, and phrase representations. DIORA achieves multiple new state-of-the-art results for unsupervised constituency parsing (absolute improvements of **13.7%**, **11.5%**, and **7.8%** on WSJ, WSJ-40, and MultiNLI), has a greater recall on more constituent types than a strong baseline, and produces meaningful phrase representations.

2 DIORA: Deep Inside-Outside Recursive Autoencoders

Our goal is to design a model and unsupervised training procedure that learns structure from raw text. The design of DIORA is based on our hypothesis is that the most effective compression of a sentence will be derived from following the true syntactic structure of the underlying input. Our approach builds on previous latent tree chart parsers which are augmented with the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) and trained to reproduce each input word from its outside context. Based on our hypothesis, loosely inspired by the linguistic "substitution principle" (Frege, 1960), the model will best reconstruct the input by discovering and exploiting syntactic regularities of the text.

The inside pass of our method recursively compresses the input sequence, at each step inputting the vector representations of the two children into a composition function (§2.1.1) that outputs an inside vector representation of the parent. This process continues up to the root of the tree, eventually yielding a single vector representing the entire sentence (Figure 2a). This is loosely analogous to the compression step of an autoencoder and equivalent to existing latent tree chart parsers forward pass (Maillard et al., 2017). Following this, we initiate the outside pass of our algorithm with a generic (root) representation that is learned as a separate parameter. As the outside step of the inside-outside algorithm (Figure 2b), we unfold until finally producing representations of the leaf nodes. These leaves are then optimized to reconstruct the input sentence as done in an autoencoder-based deep neural network.

2.1 Filling the Chart with Inside-Outside

Each inside representation is the root of a particularly sub-tree, and that representation is generated by considering only the descendant constituents within that sub-tree, ignoring any outside context. After the inside representations are calculated, we

perform a top-down outside pass to compute outside representations. The outside representations are encoded by looking at only the context of a given sub-tree. Once the chart is filled, each constituent k (cell in the chart) is associated with an inside vector $\bar{a}(k)$, an outside vector $\bar{b}(k)$, inside compatibility score $\bar{e}(k)$ and outside compatibility score $\bar{f}(k)$.

The input to our model is a sentence $\boldsymbol{x}$ made up of T tokens, $x_0, x_1, ..., x_{T-1}$. Each token x_i has a corresponding pre-trained embedded vector v_i.

2.1.1 Inside Pass

For each pair of neighboring constituents i and j [1], we compute a *compatibility* score and a *composition* vector. The score and vector that represent a particular span k are computed using a soft weighting over all possible pairs of constituents, that together fully cover the span (we refer to this set of constituent pairs as $\{k\}$).

Vectors for spans of length 1 are initialized as a non-linear transformation [2] of the embedded input v_i, and the scores associated with these spans are set to 0:

$$\begin{bmatrix} x \\ o \\ u \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} (U_\psi v_k + b)$$
$$\bar{a}(k) = o + \tanh(x \odot u)$$
$$\bar{e}(k) = 0$$

Higher levels of the chart are computed as a weighted summation of constituent pairs:

$$\bar{a}(k) = \sum_{i,j \in \{k\}} e(i,j)\, a(i,j)$$
$$\bar{e}(k) = \sum_{i,j \in \{k\}} e(i,j)\, \hat{e}(i,j)$$

The compatibility function $\hat{e}$ is meant to produce a score for how likely a pair of neighboring cells are to be merged. We implement this as a bilinear function of the vectors from neighboring spans, using a learned parameter matrix S. We additionally add the individual scores from each two merging cells. Intuitively, these individual scores correspond to how likely each of the cells would exist in the final binary tree independently. The formula for the compatibility function (and its normalized form e) is defined as follows:

$$e(i,j) = \frac{\exp(\hat{e}(i,j))}{\sum_{\hat{i},\hat{j} \in \{k\}} \exp(\hat{e}(\hat{i},\hat{j}))}$$
$$\hat{e}(i,j) = \phi(\bar{a}(i), \bar{a}(j); S_\alpha) + \bar{e}(i) + \bar{e}(j)$$

Where the bilinear projection ϕ is defined as:

$$\phi(u, v; W) = u^\top W v$$

For the composition function a we used either a TreeLSTM (Tai et al., 2015) or a 2-layer MLP (see Appendix A.2 for more precise definitions on both methods). In order for the remainder of equations to remain agnostic to the choice of composition function, we refer to the function as Compose, which produces a hidden state vector h and, in the case of TreeLSTM, a cell state vector c, resulting in:

$$a(i,j) = \text{Compose}_\alpha(\bar{a}(i), \bar{a}(j))$$

2.1.2 Outside Pass

The outside computation is similar to the inside pass (depicted in Figure 2b).

The root node of the outside chart is learned as a bias. Descendant cells are predicted using a disambiguation over the possible outside contexts. Each component of the context consists of a sibling cell from the inside chart and a parent cell from the outside chart.

The function f is analogous to the function e. It is normalized over constituent pairs i, j for the span k, and is used to disambiguate among the many outside contexts. The function b generates a phrase representation for the missing sibling cell. Equations for the outside computation follow:

$$\bar{b}(k) = \sum_{i,j \in \{k\}} f(i,j)\, b(i,j)$$
$$\bar{f}(k) = \sum_{i,j \in \{k\}} f(i,j)\, \hat{f}(i,j)$$
$$b(i,j) = \text{Compose}_\beta(\bar{a}(i), \bar{b}(j))$$
$$\hat{f}(i,j) = \phi(\bar{a}(i), \bar{b}(j); S_\beta) + \bar{e}(i) + \bar{f}(j)$$

In the majority of our experiments, the Compose used in b shares parameters with a used in the inside pass, as do the compatibility functions $\hat{e}$ and $\hat{f}$ (see §3.4 for results on the effects of parameter sharing).

[1] The symbols i, j, and k are identifiers of spans from the input $\boldsymbol{x}$. The symbol i^* identifies a token from the set of negative examples $\{\boldsymbol{x}^*\}$.

[2] This function shares its bias term b with Compose_α, although U_ψ is not tied to any other weights.

2.2 Training Objective

To train our model we use an autoencoder-like language modeling objective. In a standard autoencoder, the entire input $\boldsymbol{x}$ is compressed into a single lower dimensional representation. This representation, $\boldsymbol{z}$, is then decompressed and trained to reconstruct $\boldsymbol{x}$. In our model, we never condition the reconstruction of $\boldsymbol{x}$ on a single $\boldsymbol{z}$ because the root's outside representation is initialized with a bias rather than the root's own inside vector. Instead, we reconstruct $\boldsymbol{x}$ conditioned on the many sub-tree roots, each of which is only a compression of a *subset* of the input.

To approximate this reconstruction we use a max-margin loss considering a set $\{\boldsymbol{x}^*\}$ of N negative examples that are sampled according to their frequency from the vocabulary (further details in Appendix A.1). The terminal outside vector $\bar{b}(i)$ is trained to predict its original input v_i.

The per-instance loss function is described in Equation 1:

$$L_{\boldsymbol{x}} = \sum_{i=0}^{T-1} \sum_{i^*=0}^{N-1} \max(0, 1 - \bar{b}(i) \cdot \bar{a}(i) + \bar{b}(i) \cdot \bar{a}(i^*)) \quad (1)$$

The max-margin loss does not provide a gradient if the predicted vector is closer to its ground truth than the negative example by a margin greater than 1. For that reason, we also experimented with an objective based on cross-entropy, described in Equation 2:

$$Z^* = \sum_{i^*=0}^{N-1} \exp(\bar{b}(i) \cdot \bar{a}(i^*))$$
$$L_{\boldsymbol{x}} = -\sum_{i=0}^{T-1} \log \frac{\exp(\bar{b}(i) \cdot \bar{a}(i))}{\exp(\bar{b}(i) \cdot \bar{a}(i)) + Z^*} \quad (2)$$

2.3 DIORA CKY Parsing

To obtain a parse with DIORA, we populate an inside and outside chart using the input sentence. We can extract the maximum scoring parse based on our single grammar rule using the CKY procedure (Kasami, 1966; Younger, 1967). The steps for this procedure are described in Algorithm 1 and its runtime complexity in Appendix A.4.

Algorithm 1 Parsing with DIORA

procedure CKY(chart)
  Initialize terminal values.
  for each $k \in$ chart $\mid$ SIZE$(k) = 1$ **do**
    $x_k \leftarrow 0$
  Calculate a maximum score for each span, and record a backpointer.
  for each $k \in$ chart **do**
    $x_k \leftarrow \max_{i,j \in \{k\}} [x_i + x_j + e(i,j)]$
    $\pi_k^i, \pi_k^j \leftarrow \arg\max_{i,j \in \{k\}} [x_i + x_j + e(i,j)]$
  Backtrack to get the maximal tree.
  procedure BACKTRACK(k)
    if SIZE$(k) = 1$ **then**
      return k
    $i \leftarrow$ BACKTRACK(π_k^i)
    $j \leftarrow$ BACKTRACK(π_k^j)
    return (i, j)
  return BACKTRACK$(k \leftarrow$ root$)$

3 Experiments

To evaluate the effectiveness of DIORA, we run experiments on unsupervised parsing, unsupervised segment recall, and phrase similarity. The model has been implemented in PyTorch (Team, 2018) and the code is published online.[3] For training details, see Appendix A.1.

3.1 Unsupervised Parsing

We first evaluate how well our model predicts a full unlabeled constituency parse. We look at two data sets used in prior work (Htut et al., 2018), The Wall Street Journal (WSJ) section of Penn Treebank (Marcus et al., 1993), and the automatic parses from MultiNLI (Williams et al., 2018b). WSJ has gold human-annotated parses and MultiNLI contains automatic parses derived from a supervised parser (Manning et al., 2014).

In addition to PRPN (Shen et al., 2018),[4] we compare our model to deterministically constructed left branching, right branching, balanced, and random trees. We also compare to ON-LSTM (Shen et al., 2019), an extension of the PRPN model, RL-SPINN (Yogatama et al., 2017), an unsupervised shift-reduce parser, and ST-Gumbel (Choi et al., 2018), an unsupervised chart parser. The latter two of these models are trained to predict the downstream task of natural language inference (NLI).

[3] https://github.com/iesl/diora

[4] We consider the PRPN models using LM stopping criteria, which outperformed UP.

3.1.1 Binarized WSJ and MultiNLI results

For the full WSJ test set and MultiNLI datasets we follow the experimental setup of previous work (Williams et al., 2018a). We binarize target trees using Stanford CoreNLP (Manning et al., 2014) and do not remove punctuation (experiments in §3.1.2 do remove punctuation).

Latent tree models have been shown to perform particularly poorly on attachments at the beginning and end of the sequence (Williams et al., 2018a). To address this, we incorporate a post-processing heuristic (denoted as +PP in result tables)[5]. This heuristic simply attaches trailing punctuation to the root of the tree, regardless of its predicted attachment.

In Table 1, we see that DIORA^{+PP} achieves the highest average and maximum F1 from five random restarts. This model achieves a mean F1 7 points higher than ON-LSTM and an increase of over 6.5 max F1 points. We also see that DIORA exhibits much less variance between random seeds than ON-LSTM. Additionally, we find that PRPN-UP and DIORA benefit much more from the +PP heuristic than PRPN-LM. This is consistent with qualitative analysis showing that DIORA and PRPN-UP incorrectly attach trailing punctuation much more often than PRPN-LM.

On the MultiNLI dataset, PRPN-LM is the top performing model without using the +PP heuristic while DIORA matches PRPN-UP (Table 2. Using the heuristic, DIORA greatly surpasses both variants of PRPN. However, it is worth noting that this is not a gold standard evaluation and instead evaluates a model's ability to replicate the output of a trained parser (Manning et al., 2014). A second caveat is that SNLI (Bowman et al., 2015) and MultiNLI contain several non-newswire domains. Syntactic parsers often suffer significant performance drops when predicting outside of the newswire domain that the models were trained on.

3.1.2 WSJ-10 and WSJ-40 results

We also compare our models to two subsets of the WSJ dataset that were used in previous unsupervised parsing evaluations. WSJ-10 and WSJ-40 contain sentences up to length 10 and 40 respectively after punctuation removal. We do not binarize either of these two splits in order to compare to previous work (see Appendix A.3 for more details on WSJ split differences). Not binarizing the target trees sets an upper-bound on the performance of our models, denoted as UB in Table 3.

We compare against previous notable models for this task: CCM (Klein and Manning, 2002) uses the EM algorithm to learn probable nested

[5]We did not have access to predictions or an implementation of the concurrent ON-LSTM model and therefore could not apply the +PP heuristic.

Model	$F1_{\mu}$	$F1_{max}$	δ
LB	13.1	13.1	12.4
RB	16.5	16.5	12.4
Random	21.4	21.4	5.3
Balanced	21.3	21.3	4.6
RL-SPINN†	13.2	13.2	-
ST-Gumbel - GRU†	22.8 ±1.6	25.0	-
PRPN-UP	38.3 ±0.5	39.8	5.9
PRPN-LM	35.0 ±5.4	42.8	6.2
ON-LSTM	47.7 ±1.5	49.4	5.6
DIORA	48.9 ±0.5	49.6	8.0
PRPN-UP^{+PP}	-	45.2	6.7
PRPN-LM^{+PP}	-	42.4	6.3
DIORA^{+PP}	**55.7** ±0.4	**56.2**	8.5

Table 1: Full WSJ (test set) unsupervised unlabeled binary constituency parsing including punctuation. † indicates trained to optimize NLI task. Mean and max are calculated over five random restarts. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). The depth (δ) is the average tree height. +PP refers to post-processing heuristic that attaches trailing punctuation to the root of the tree. The top F1 value in each column is bolded.

Model	$F1_{median}$	$F1_{max}$	δ
Random	27.0	27.0	4.4
Balanced	21.3	21.3	3.9
PRPN-UP	48.6	-	4.9
PRPN-LM	50.4	-	5.1
DIORA	51.2	53.3	6.4
PRPN-UP^{+PP}	-	54.8	5.2
PRPN-LM^{+PP}	-	50.4	5.1
DIORA^{+PP}	59.0	**59.1**	6.7

Table 2: NLI unsupervised unlabeled binary constituency parsing comparing to CoreNLP predicted parses. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). F1 median and max are calculated over five random seeds and the top F1 value in each column is bolded. Note that we use median rather than mean in order to compare with previous work.

bracketings over a sentence using gold or induced part-of-speech tags, and PRLG (Ponvert et al., 2011) performs constituent parsing through consecutive rounds of sentence chunking.

In Table 3, we see that DIORA outperforms the previous state of the art for WSJ-40, PRLG, in max F1. The WSJ-10 split has been difficult for latent tree parsers such as DIORA, PRPN, and ON-LSTM, none of which (including our model) are able to improve upon previous non-neural methods. However, when we compare trends between WSJ-10 and WSJ-40, we see that DIORA does a better job at extending to longer sequences.

	WSJ-10		WSJ-40	
Model	**F1**$_\mu$	**F1**$_{max}$	**F1**$_\mu$	**F1**$_{max}$
UB	87.8	87.8	85.7	85.7
LB	28.7	28.7	12.0	12.0
RB	61.7	61.7	40.7	40.7
CCM†	-	63.2	-	-
CCM$_{gold}$†	-	71.9	-	33.7
PRLG †	-	**72.1**	-	54.6
PRPN$_{NLI}$	66.3 $\pm$0.8	68.5	-	-
PRPN‡	70.5 $\pm$0.4	71.3	-	52.4
ON-LSTM‡	65.1 $\pm$1.7	66.8	-	-
DIORA	67.7 $\pm$0.7	68.5	60.6 $\pm$0.2	**60.9**

Table 3: WSJ-10 and WSJ-40 unsupervised non-binary unlabeled constituency parsing with punctuation removed. † indicates that the model predicts a full, non-binary parse with additional resources. ‡ indicates model was trained on WSJ data and PRPN$_{NLI}$ was trained on MultiNLI data. CCM uses predicted POS tags while CCM$_{gold}$ uses gold POS tags. PRPN F1 was calculated using the parse trees and results provided by Htut et al. (2018). LB and RB are the left and right-branching baselines. UB is the upper bound attainable by a model that produces binary trees.

3.2 Unsupervised Phrase Segmentation

In many scenarios, one is only concerned with extracting particular constituent phrases rather than a full parse. Common use cases would be identifying entities, noun phrases, or verb phrases for downstream analysis. To get an idea of how well our model can perform on phrase segmentation, we consider the maximum recall of spans in our predicted parse tree. We leave methods for cutting the tree to future work and instead consider the maximum recall of our model which serves as an upper bound on its performance. Recall here is the percentage of labeled constituents that appear in our predicted tree relative to the total number of constituents in the gold tree. These scores are separated by type and presented in Table 4.

In Table 4 we see the breakdown of constituent recall across the 10 most common types. DIORA achieves the highest recall across the most types and is the only model to perform effectively on verb-phrases. Interestingly, DIORA performs worse than PRPN-LM at prepositional phrases.

3.3 Phrase Similarity

One of the goals of DIORA is to learn meaningful representations for spans of text. Most language modeling methods focus only on explicitly modeling token representations and rely on ad-hoc post-processing to generate representations for longer spans, typically relying on simple arithmetic functions of the individual tokens.

To evaluate our model's learned phrase representations, we look at the similarity between spans of the same type within labeled phrase datasets. We look at two datasets. CoNLL 2000 (Tjong Kim Sang and Buchholz, 2000) is a shallow parsing dataset containing spans of noun phrases, verb phrases, etc. CoNLL 2012 (Pradhan et al., 2012) is a named entity dataset containing 19 different entity types.

For each of the labeled spans with length greater than one, we first generate its phrase representation. We then calculate its cosine similarity to all other labeled spans. We then calculate if the label for that query span matches the labels for each of the K most similar other spans in the dataset. In Table 5 we report precision@K for both datasets and various values of K.

The first baseline we compare against produces phrase representations from averaging context-insensitive (CI) ELMo vectors of individual tokens with the span. The second uses sentence-insensitive (SI) ELMo vectors, running the full ELMo over only the relevant tokens and ignoring the rest of the sentence. We also look at ELMo's output when given the entire sentence. When analyzing our baselines that run the full ELMo, we follow the procedure described in (Peters et al., 2018b) and represent phrases as a function of its first and last hidden state. We extract these states from the final ELMo layer (3rd BiLSTM) as these consistently gave the best performance among other options. For DIORA, we use the concatenation of the inside and outside representations ($[\bar{a}; \bar{b}]$).

Label	Count	DIORA	P-UP	P-LM
NP	297,872	**0.767**	0.687	0.598
VP	168,605	**0.628**	0.393	0.316
PP	116,338	0.595	0.497	**0.602**
S	87,714	**0.798**	0.639	0.657
SBAR	24,743	**0.613**	0.403	0.554
ADJP	12,263	**0.604**	0.342	0.360
QP	11,441	**0.801**	0.336	0.545
ADVP	5,817	**0.693**	0.392	0.500
PRN	2,971	**0.546**	0.127	0.144
SINV	2,563	0.926	0.904	**0.932**

Table 4: Segment recall from WSJ separated by phrase type. The 10 most frequent phrase types are shown above, and the highest value in each row is bolded. P-UP=PRNP-UP, P-LM=PRPN-LM

For CoNLL 2000, we find that our model outperforms all baselines for all values of K. This demonstrates DIORA's ability to capture and represent syntactic information within phrases. For CoNLL 2012, we find that DIORA outperforms both ELMo$_{CI}$ and ELMo$_{SI}$ while ELMo performs best overall. ELMo$_{CI}$ is surprisingly effective on this dataset even though it performed more poorly on CoNLL 2000. These results indicate that DIORA is capturing syntax quite well, but still has room to improve on more fine-grained semantic representations.

3.4 Impact of Modeling Choices

To test the impact of our modeling choices, we compared the performance of two different losses and four different composition functions on the full WSJ validation set. The losses were covered in Equations 1 (Margin) and 2 (Softmax). The two primary methods of composition we considered were TreeLSTM (Tai et al., 2015) and MLP (a 2-hidden layer neural network). In addition, we experimented with a simple *kernel* of the MLP input $[x; y; x \odot y; x - y]$ and with a setting where both the inside and outside parameters are *shared*.

The results are shown in Table 6. We see that MLP composition consistently performs better than with TreeLSTM, that MLP benefits from the Softmax loss, and that the best performance comes from sharing parameters. All other experimental results use this highly performant setting unless otherwise specified.

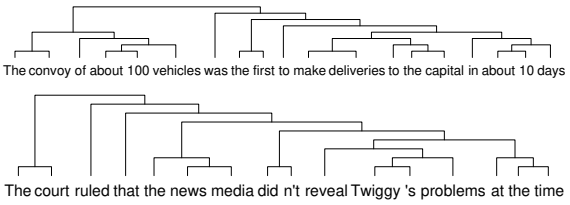

Figure 3: DIORA can match the ground truth exactly.

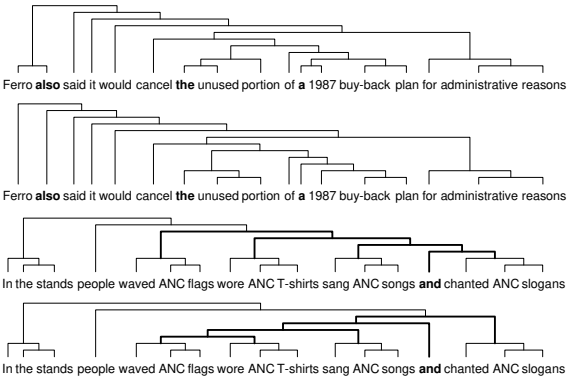

Figure 4: At times, DIORA exhibits contrary behavior to the ground truth inevitably leading to some error. DIORA's output is shown above the ground truth.[6]

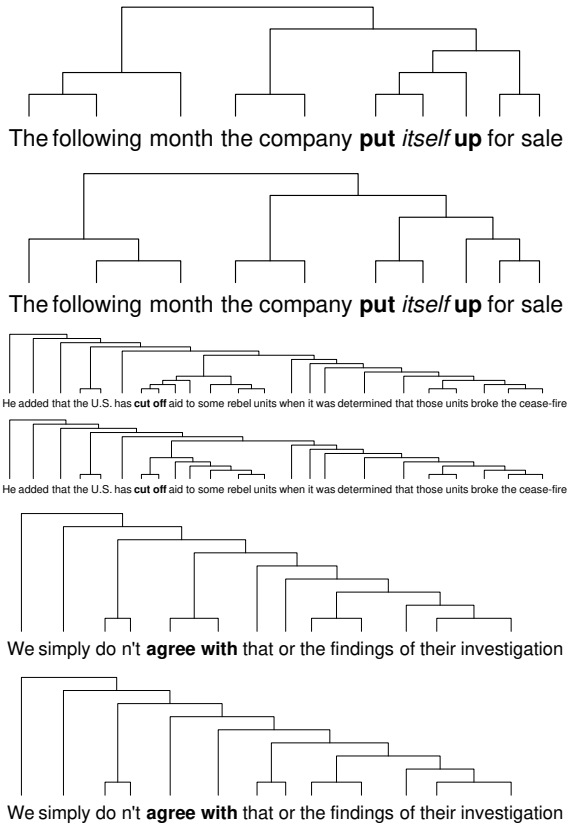

Figure 5: DIORA often groups verbs and particles (top), sometimes exactly as the ground truth (middle). Occasionally, errors are particle-like (bottom). DIORA's output is shown above the ground truth.[6]

[6]Ground truth parses are binarized unless otherwise specified. All examples of DIORA parses are already binary. Some punctuation has been removed for easier readability.

Model	Dim	CoNLL 2000			CoNLL 2012		
		P@1	P@10	P@100	P@1	P@10	P@100
Random	800	0.684	0.683	0.680	0.137	0.133	0.135
ELMo$_{CI}$	1024	0.962	0.955	0.957	0.708	0.643	0.544
ELMo$_{SI}$	4096	0.970	0.964	0.955	0.660	0.624	0.533
ELMo	4096	0.987	0.983	0.974	**0.896**	**0.847**	**0.716**
DIORA$_{In/Out}$	800	**0.990**	**0.985**	**0.979**	0.860	0.796	0.646

Table 5: P@1, P@10, and P@100 for labeled chunks from CoNLL-2000 and CoNLL 2012 datasets. For all metrics, higher is better. The top value in each column is bolded. Diora uses the concatenation of the inside and outside vector at each cell which performed better than either in isolation.

3.5 Qualitative Results

Looking at our model's output, we see that some trees are an exact replication of the binarized ground truth (Fig. 3), or very close (Fig. 4). For future work we intend to explore common patterns in DIORA's learned structure, although some patterns are already recognizable, such as the affinity to group particles and verbs (Fig. 5).

4 Related Work

Latent Tree Learning A brief survey of neural latent tree learning models was covered in (Williams et al., 2018a). The first positive result for neural latent tree parsing was shown in (Htut et al., 2018), which used a language modeling objective. The model in (Liu et al., 2018) uses an inside chart and an outside procedure to calculate marginal probabilities in order to align spans between sentences in entailment.

Composition	Loss	F1$_\mu$	
		∅	+PP
TreeLSTM	Margin	49.9	53.1
TreeLSTM	Softmax	52.0	52.9
MLP	Margin	49.7	54.4
MLP	Softmax	52.6	55.5
MLP$_{\text{Kernel}}$	Softmax	51.8	54.8
MLP$_{\text{Shared}}$	Softmax	50.8	56.7

Table 6: F1 for different model variants on the binary WSJ validation set with included punctuation. The binary trees are as-is (∅) or modified according to the post-processing heuristic ($+PP$). The mean F1 is shown across three random seeds.

Neural Inside-Outside Parsers The Inside-Outside Recursive Neural Network (IORNN) (Le and Zuidema, 2014) is closest to ours. It is a graph-based dependency parser that uses beam search and can reliably find accurate parses when retaining a k-best list. In contrast, our model produces the most likely parse given the learned compatibility of the constituents. The Neural CRF Parser (Durrett and Klein, 2015), similar to DIORA, performs exact inference on the structure of a sentence, although requires a set of grammar rules and labeled parse trees during training. DIORA, like Liu et al. (2018), has a single grammar rule that applies to any pair of constituents and does not use structural supervision.

Learning from Raw Text Unsupervised learning of syntactic structure has been an active research area (Brill et al., 1990), including for unsupervised segmentation (Ando and Lee, 2000; Goldwater et al., 2009; Ponvert et al., 2011) and unsupervised dependency parsing (Spitkovsky et al., 2013). Some models exploit the availability of parallel corpora in multiple languages (Das and Petrov, 2011; Cohen et al., 2011). Others have shown that dependency parsing can be used for unsupervised constituency parsing (Spitkovsky et al., 2013; Klein and Manning, 2004), or that it's effective to prune a random subset of possible trees (Bod, 2006). These approaches aren't necessarily orthogonal to DIORA. For instance, our model may benefit when combined with an unsupervised dependency parser.

5 Conclusion

In this work we presented DIORA, an unsupervised method for inducing syntactic trees and representations of constituent spans. We showed

inside-outside representations constructed with a latent tree chart parser and trained with an autoencoder language modeling objective learns syntactic structure of language effectively. In experiments on unsupervised parsing, chunking, and phrase representations we show our model is comparable to or outperforms previous methods, achieving the state-of-the-art performance on unsupervised unlabeled constituency parsing for the full WSJ (with punctuation), WSJ-40, and NLI datasets. We also show our model obtains higher segment recall than a comparable model and outperforms strong baselines on phrase representations on a chunking dataset.

While the current model seems to focus primarily on syntax, future work can improve the model's ability to capture fine-grained semantics. Potential avenues include training larger models over much larger corpora, extra unsupervised or weakly-supervised phrase classification objectives, and other modeling enhancements. We are also eager to apply DIORA to other domains and languages which do not have rich linguistically annotated training sets.

Acknowledgements

We are grateful to Carolyn Anderson, Adina Williams, Phu Mon Htut, and 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 National Science Foundation (NSF) grant numbers DMR-1534431, IIS-1514053 and CNS-0958392. 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.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Association for Computational Linguistics (ACL)*.

Rie Kubota Ando and Lillian Lee. 2000. Mostly-unsupervised statistical segmentation of japanese: Applications to kanji. In *North American Association for Computational Linguistics (NAACL)*.

James K Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132.

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

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)*.

Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. 1990. Deducing linguistic structure from the statistics of large corpora. In *Information Technology, 1990.'Next Decade in Information Technology', Proceedings of the 5th Jerusalem Conference on (Cat. No. 90TH0326-9)*, pages 380–389. IEEE.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Association for Computational Linguistics (ACL)*.

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

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Association for Computational Linguistics (ACL)*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Association for Computational Linguistics (ACL)*.

Friedrich Ludwig Gottlob Frege. 1960. On sense and reference. In *Zeitschrift für Philosophie und philosophische Kritik 100 (1892) 25-50; translated in Translations from the Philosophical Writings of Gottlob Frege (ed. by P. Geach and M. Black)*. Oxford.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Joint workshop on unsupervised and semi-supervised learning in NLP*. Association for Computational Linguistics (ACL).

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2009. A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112:21–54.

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)*.

Phu Mon Htut, Kyunghyun Cho, and Samuel R Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Empirical Methods in Natural Language Processing (EMNLP): Short Paper*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Association for Computational Linguistics (ACL)*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Association for Computational Linguistics (ACL)*.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Phong Le and Willem Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1155–1164.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured alignment networks for matching sentences. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *arXiv preprint arXiv:1705.09189*.

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.

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

Matthew E Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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)*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*.

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)*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Conference on Computational Natural Language Learning (CoNLL)*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations

from tree-structured long short-term memory networks. In *Association for Computational Linguistics (ACL)*.

Pytorch Core Team. 2018. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. http://pytorch.org/. Accessed: 2018-09-26.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

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)*.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Poorya Zaremoodi and Gholamreza Haffari. 2018. Incorporating syntactic uncertainty in neural machine translation with a forest-to-sequence model. In *International Conference on Computational Linguistics*.

A Appendices

A.1 Training Details

Training Data. Sentences of length ≤ 20 from the SNLI and MultiNLI training sets.

Optimization. We train our model using stochastic gradient descent with the Adam optimization algorithm (Kingma and Ba, 2014). Cells were normalized to have magnitude of 1, following Socher et al. (2011a). For instance, $\bar{a}(k) := \bar{a}(k)/\left\|\bar{a}(k)\right\|^2$. Gradients are clipped to a maximum L2-norm of 5.

Hyperparameters. Chosen using grid search over cell-dimension $\{400\text{D}, 800\text{D}\}$ and learning rate $\{2, 4, 8, 10, 20\} \cdot 10^{-4}$.

Early Stopping. Using unlabeled parsing F1 against the binarized WSJ validation set.

Vocabulary. The model is trained in an open-vocabulary setting using pre-trained context-insensitive character embeddings. The embedder is taken from ELMo (Peters et al., 2018a).

Batching. Batches were constructed such that they contained sentences of uniform length. Using batch size 128 for 400D and 64 for 800D.

Sampling. N negatives are sampled for each batch. All experiments use $N = 100$.

Training Steps. 1M parameter updates, taking 3 days using 4x Nvidia 1080ti.

A.2 Composition and Input Transform

TreeLSTM. The TreeLSTM (Tai et al., 2015) function produces a hidden state vector h and cell state vector c given two input vectors h_i and h_j.

$$\begin{bmatrix} x \\ f_i \\ f_j \\ o \\ u \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(U \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b + \begin{bmatrix} 0 \\ \omega \\ \omega \\ 0 \\ 0 \end{bmatrix} \right)$$

$$c = c_i \odot f_i + c_j \odot f_j + x \odot u$$

$$h = o + \tanh(c)$$

The constant ω is set to 1 for the inside, 0 for the outside. U and b are learned.

MLP. MLP (Multi-Layer Perceptron) is a deep non-linear composition with the following form:

$$h = W_1 \left(W_0 \left\langle h_i, h_j \right\rangle + b \right) + b_1$$

The operator $\langle h_i, h_j \rangle$ is a concatenation $[h_i; h_j]$. For the MLP^{Kernel} $\langle h_i, h_j \rangle$ is more involved to support further interaction between the two input vectors $[h_i; h_j; h_i \odot h_j; h_i - h_j]$. The variables W_0, W_1, b, b_1 are learned and c is unused.

A.3 Reproducing Parsing Results

In Table 7, we've organized a reference for creating various splits of the WSJ for the purpose of evaluating unsupervised parsing. Some splits use only the test set (section 23), others use all of the training, validation, and test data. Optionally, punctuation is stripped and sentences greater than a specified length are ignored. Predictions can be compared to the full parse trees in the annotated data, or to a binarized version. The PARSEVAL specification calculated bracketing F1 considering all spans, although some previous work diverts from PARSEVAL and ignores spans that are trivially correct (ones over the entire sentence).

	WSJ	WSJ-10	WSJ-40
Split	Test	All	Test
w/ Punctuation	Yes	No	No
Max Length	∞	10	40
Binarized	Yes	No	No
Trivial Spans	Yes	No	No

Table 7: Settings for unlabeled binary bracketing evaluation for different splits of the WSJ corpus.

A.4 Runtime Complexity

The runtime complexities for DIORA's methods are shown in Table 8. The parallel column represents the complexity when the values for all constituent pairs are computed simultaneously, assuming that these computations are independent and do not depend on values that have yet to be computed. Linear complexity is theoretically feasible depending on batch size, input length, and number of computational cores. In practice, one might experience super-linear performance.

Although both the inside pass and outside pass have an upper bound of n^3 operations, the outside pass will have more operations than the inside pass for sentences of length > 1.

As a point of reference, our implementation computes the loss over the entire WSJ corpus in 5 minutes 30 seconds at a rate of 3,500 words per second using a single GPU.

Method	Serial	Parallel
Inside Pass	$O(n^3)$	$O(n)$
Outside Pass	$O(n^3)$	$O(n)$
Training Objective	$O(n \cdot N)$	$O(n)$
CKY	$O(n^3)$	$O(n)$

Table 8: Runtime complexity for methods associated with DIORA in terms of sentence length n and number of negative examples per token N. Each column represents the complexity when the values for each constituent are computed serially or in parallel.

A.5 Parse Trees

Examples of parse trees derived from the compatibility scores are shown in Figures 6 and 7. Some punctuation has been removed for readability.

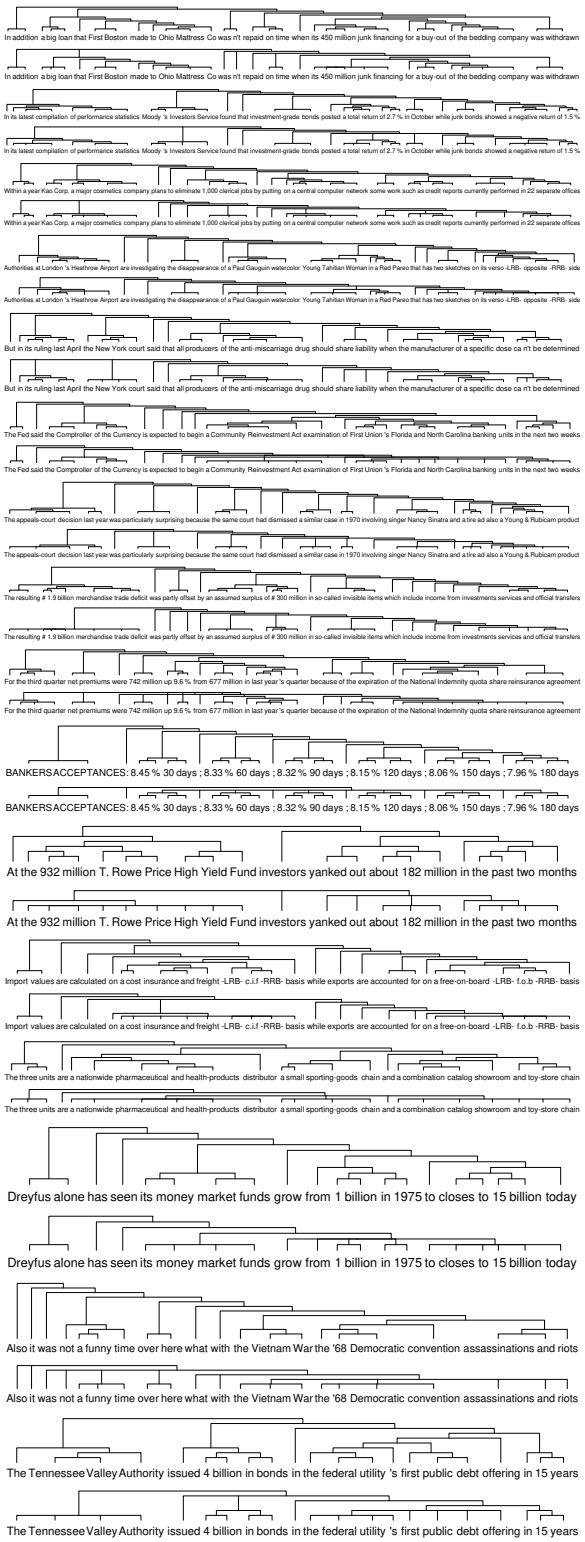

Figure 6: Examples where DIORA achieves 100% recall compared with the raw (n-ary) ground truth, but less than 100% accuracy on the binarized ground truth. DIORA's output is shown above the ground truth.

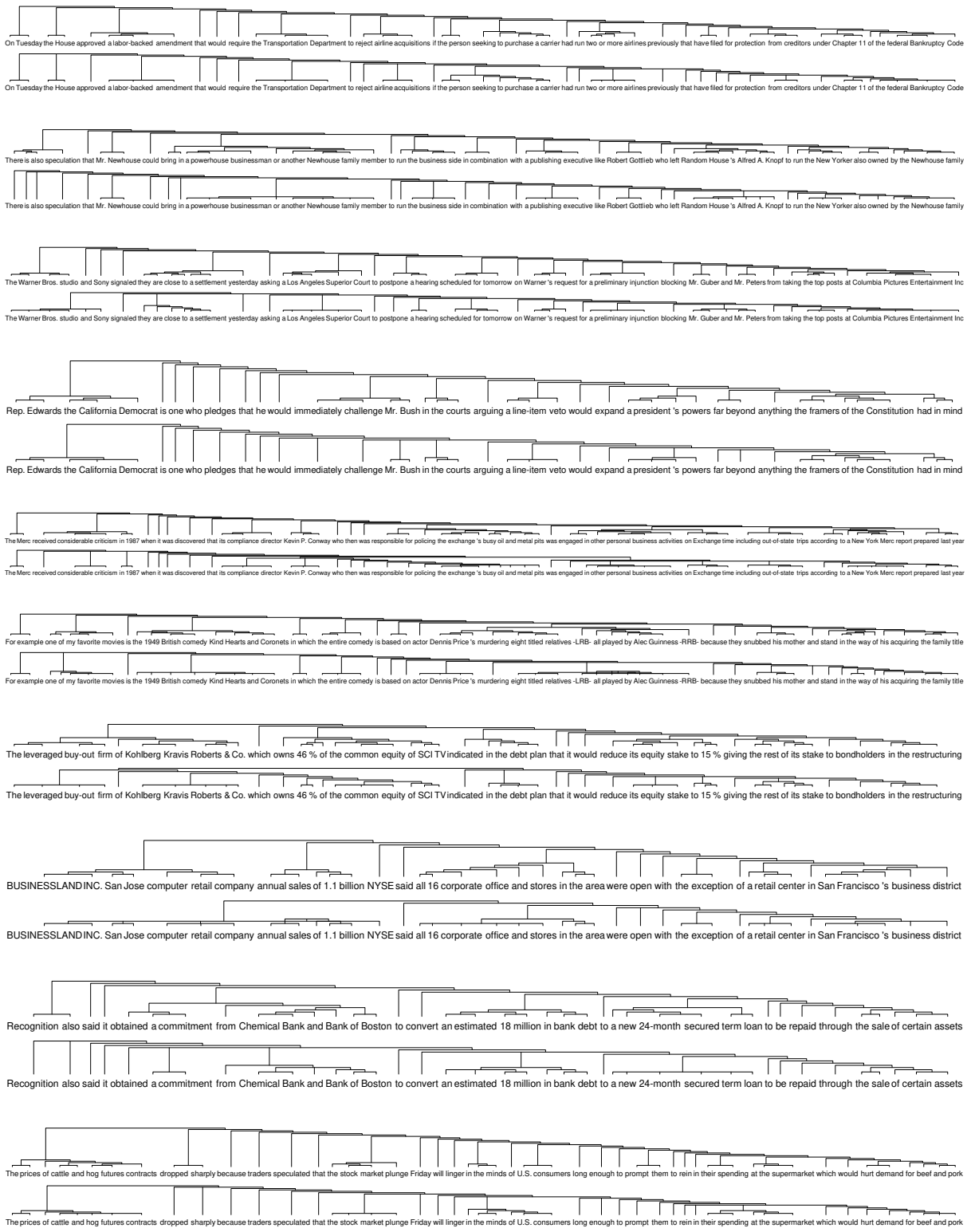

Figure 7: DIORA can perform close to the ground truth even on long sentences. In this figure, n-ary trees are shown for the ground truth. DIORA's output is shown above the ground truth.