UNSUPERVISED LATENT TREE INDUCTION WITH DEEP INSIDE-OUTSIDE RECURSIVE AUTO-ENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Syntax is a powerful abstraction for language understanding. Many downstream tasks require segmenting input text into meaningful constituent chunks (e.g., noun phrases or entities); more generally, models for learning semantic representations of text benefit from integrating syntax in the form of parse trees (e.g., tree-LSTMs). Supervised parsers have traditionally been used to obtain these trees, but lately interest has increased in unsupervised methods that induce syntactic representations directly from unlabeled text. To this end, we propose the deep inside-outside recursive autoencoder (DIORA), a fully-unsupervised method for discovering syntax that simultaneously learns representations for constituents within the induced tree. Unlike many prior approaches, DIORA does not rely on supervision from auxiliary downstream tasks and is thus not constrained to particular domains. Furthermore, competing approaches do not learn explicit phrase representations along with tree structures, which limits their applicability to phrase-based tasks. Extensive experiments on unsupervised parsing, segmentation, and phrase clustering demonstrate the efficacy of our method. DIORA achieves the state of the art in unsupervised parsing (46.9 F1) on the benchmark WSJ dataset.

1 INTRODUCTION

Syntax in the form of parse trees is an essential component of many natural language processing tasks. Constituent spans taken from a parse tree are useful for tasks such as relation extraction Verga et al. (2016) and semantic role labeling (Strubell et al., 2018), while the full parse itself can be used to build higher-quality systems for machine translation (Aharoni and Goldberg, 2017) and text classification (Tai et al., 2015). Supervised parsers trained on datasets such as the Penn Treebank (Marcus et al., 1994) are traditionally used to obtain these trees; however, these datasets are generally small and restricted to the newswire domain. For out-of-domain applications, it is generally infeasible to create new treebanks, as syntactic annotation is expensive and time-consuming.

Motivated by these limitations, we propose a method that extracts both shallow parses (i.e., noun phrases or entities) and full syntactic trees from any domain or language automatically *without any training data*. In addition to just producing the parse, we want our model to build representations for internal constituents that obey syntactic and semantic regularities, as we can then easily inject these representations into downstream tasks. Our model extends existing work on latent tree chart parsers (Le and Zuidema, 2015; Yogatama et al., 2016; Maillard et al., 2017; Choi et al., 2018), which build up representations for all internal nodes in the tree (cells in the chart) generated by a soft weighting over all possible sub-trees (Section 2).

In previous work, the representation at the root node is used as a sentence encoding and trained to optimize some downstream task, typically natural language inference. Unfortunately, this method requires sentence level annotations to train the model. Worse still, analysis on the trees learned by these models show that they are actually quite poor at capturing syntax that in any way resembles linguistic theory (Williams et al., 2018a). To address these issues, we incorporate the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) into a latent tree chart parser. The bottom-up inside step is equivalent to the forward-pass of previous latent tree chart parsers (Maillard et al., 2017). However, these inside representations are encoded by looking only within the current subtree, completely ignoring outside context. Thus, we perform an additional top-down outside calculation for each node in the tree incorporating external context into sub-tree representations. Finally, we train

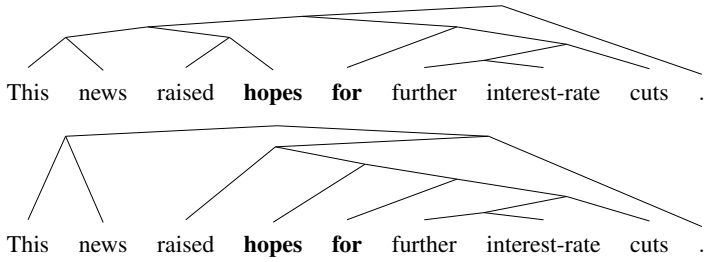


Figure 1: Example parse trees. Top PRPN-LM prediction, bottom DIORA prediction. DIORA correctly chunks the span 'raised hopes for further interest-rate cuts'.

the outside representations of leaves to reconstruct the initial input, which results in a completely unsupervised autoencoder-like objective.

Recently, Shen et al. (2018) proposed Parsing-Reading-Predict Networks (PRPN), an RNN based language model with an additional module for inferring syntactic distance. After training, this syntax module can be decomposed to recover a parse (Htut et al., 2018) via a complex mechanism that involves modeling a distribution over possible syntactic structures with a stick-breaking process. Like DIORA, this model can be trained in a completely unsupervised manner. However, it has no mechanism of explicitly modeling phrases, and span representations can only be generated by post-hoc heuristics. Additionally, finding the most probable tree in DIORA is much simpler than in PRPN, as we can just run the CKY algorithm.

To probe different properties of our model, we run experiments on unsupervised parsing, segmentation, and phrase representations. DIORA sets the state-of-the-art for unsupervised parsing on the WSJ dataset, has a greater recall on a more constituent types than PRPN, and demonstrates strong clustering of phrase representations.

2 DIORA: Deep Inside-Outside Recursive Auto-Encoder

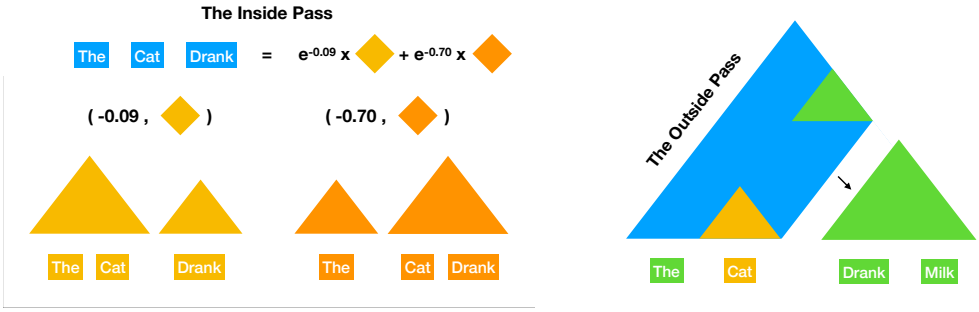
Our goal is to build an unsupervised model which can automatically discover syntactic structure from raw text. The hypothesis that our model follows is that the most efficient compression of a sentence will be derived from following the true syntactic structure of the underlying input. Our model is an extension of latent tree chart parsers augmented with the inside-outside algorithm (Baker, 1979; Lari and Young, 1990) and trained as an auto-encoder. Based on our hypothesis, the auto-encoder will best reconstruct the input by discovering and exploiting syntactic regularities of the text.

The inside phase of our method recursively compresses the input sequence into a single vector representing the sentence (Section 2.1.1). This is analogous to the compression step of an autoencoder and equivalent to existing latent tree chart parsers forward pass. Following this, we initiate the outside phase of our algorithm there a generic sentence (root) representation which is trained as a part of the model parameter. As an outside step of the inside-outside algorithm (Section 2.1.2), we expand outward until finally producing reconstructed representations of the leaf nodes. These reconstructed leaves are then optimized to reconstruct the input sentence as done in an auto-encoder based deep neural network (Section 2.2).

2.1 Filling the Chart with Inside-Outside

Each inside representation of a given sub-tree is built considering only the children constituents of that sub-tree, independent of any outside context. After the inside representations are calculated, we do a top-down outside pass to compute outside representations. The outside representations are encoded by looking at the context of a given sub-tree. In the end, each cell in the chart will contain an inside vector, inside compatibility score, outside vector, and outside compatibility score.

Once the chart is filled, each constituent k (cell in the chart) is associated with an inside vector α_k^{vec}, an outside vector β_k^{vec}, inside score α_k^{score} and outside score β_k^{score}.



(a) (b)

Figure 2: The inside and outside pass of DIORA for the input 'the cat drank milk'. a) The inside pass: The inside vector for the phrase 'the cat drank' is a weighted average of the compositions for the two possible segmentations - ((the cat), drank) and (the, (cat drank)). The weights come from the learned compatibility scores α_k^{score}. b) The outside Pass: The outside vector for the phrase 'drank milk' is a function of the outside vector of its parent and the inside vector of its sibling.

Assuming that the input to our model is a sentence X made up of T tokens, $x_0, x_1, ..., x_T$, we describe inside and outside phases of our algorithm in the following Sections 2.1.1 and 2.1.2. Also, each token x_i has a corresponding pre-trained d dimensional embedding vector v_i.

2.1.1 Inside Phase

For each pair of neighboring constituents i and j, we compute a *compatibility* score α_k^{score} and a *composition* vector α_k^{vec}. The score and vector that represents a particular span k are computed using a soft weighting over all possible pairs of constituents that together covers the span entirely (we refer to this set of constituent pairs as $\{k\}$):

Vectors for spans of length 1 are initialized using a linear transformation of the embedded input v_i. Scores associated with these spans are set to 0.

$$\alpha_k^{vec} = W_{in} v_k^T \tag{1}$$

$$\alpha_k^{score} = 0 \tag{2}$$

For higher levels of the chart we use:

$$\alpha_k^{vec} = \sum_{i,j \in \{k\}} e^{compat(i,j)} compose(i,j) \tag{3}$$

$$\alpha_k^{score} = \sum_{i,j \in \{k\}} e^{compat(i,j)} compat(i,j) \tag{4}$$

The compatibility function $compat$ is a bilinear function of the vectors from neighboring spans, adding their scores:

$$compat(i,j) = \alpha_i^{vec} S^{in\top} \alpha_j^{vec\top} + \alpha_i^{score} + \alpha_j^{score} \tag{5}$$

And the composition function $compose$ is a TreeLSTM (Tai et al., 2015) which produces a hidden state vector h and cell state vector c.:

$$compose(i,j) = TreeLSTM^{in}(\alpha_i^{vec}, \alpha_j^{vec}) = \begin{bmatrix} h \\ c \end{bmatrix} \tag{6}$$

Where the $TreeLSTM$ is defined as follows:

$$\begin{bmatrix} i \\ f_i \\ f_j \\ u \\ o \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(U \begin{bmatrix} h_i \\ h_j \end{bmatrix}^\top + b + \begin{bmatrix} 0 \\ \omega \\ \omega \\ 0 \\ 0 \end{bmatrix} \right) \tag{7}$$

$$c = c_i \odot \sigma(f_i) + c_j \odot \sigma(f_j) + \tanh(u) \odot \sigma(i) \tag{8}$$

$$h = \sigma(o) + \tanh(c) \tag{9}$$

The constant ω is set to 1 for the inside phase and 0 for the outside phase. The parameters U and b are not shared between the inside phase and outside phase.

2.1.2 OUTSIDE PHASE

The outside computation is similar to the inside,

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.

$$\beta_k^{vec} = \sum_{i,j \in \{k\}} e^{disamb(i,j)} predict(i,j) \tag{10}$$

$$\beta_k^{score} = \sum_{i,j \in \{k\}} e^{disamb(i,j)} disamb(i,j) \tag{11}$$

$$disamb(i,j) = \beta_i^{vec} S^{out\top} \alpha_j^{vec\top} + \beta_i^{score} + \alpha_j^{score} \tag{12}$$

$$predict(i,j) = TreeLSTM^{out}(\alpha_i^{vec}, \beta_j^{vec}) \tag{13}$$

2.2 TRAINING OBJECTIVE

To train our model we use an auto-encoder-like language modeling objective. In a standard auto-encoder, the input X is compressed into a single lower dimensional representation Y. Y is then decompressed and trained to predict X. In our model, we never condition the reconstruction of X on a single Y because the root's outside representation is initialized with a bias rather than the root's own inside vector. Instead, we reconstruct X conditioned on the many sub-tree roots, none of which is a single compression of the entire X, but rather a subset.

Each generated outside vector β_i^{vec} for constituents of length 1 are trained to predict their original input v_i. We approximate a reconstruction loss with a max-margin across N negative samples.

For each x_i, we sample N negative x_i^n uniformly at random from the vocabulary. The training objective of our model over a batch $\mathbf{B} = \{X_{T^i}, i = 1, ..., B\}$ is computed identically for all tokens (which are also all spans with length 1) within the batch and averaged to get the overall loss for the entire batch. Precisely, the loss function for each token (span k) is described in Equation 14.

$$L(\mathbf{B}) = \sum_{n=1}^{n=N} \max(0, 1 - \beta_k^{vec} * \alpha_k^{vec} + \beta_k^{vec} * \alpha_{k_n}^{vec}) \tag{14}$$

In equation 14, $\alpha_{k_n}^{vec}$ are representations for negative samples from vocabulary. Similar to input transformation, $\alpha_{k_n}^{vec}$ are also computed after applying a linear transformation over the input embeddings. As mentioned before, α_k^{vec} and β_k^{vec} are inside and outside representations, for span k, respectively.

Algorithm 1 Parsing with DIORA

procedure CKY(chart)
  for each $k \in chart \mid size(k) = 1$ **do** ▷ Initialize terminal values.
    $x_k \leftarrow 0$
  for each $k \in chart$ **do**
    $x_k \leftarrow \max_{i,j \in \{k\}} [x_i + x_j + compat(i,j)]$ ▷ Calculate a maximum score for each span.
    $b_k \leftarrow \arg\max_{i,j \in \{k\}} [x_i + x_j + compat(i,j)]$ ▷ Record a backpointer.
  procedure FOLLOW-BACKPOINTERS(k)
    if $size(k) = 1$ **then**
      return k
    $i \leftarrow$ FOLLOW-BACKPOINTERS(b_k^i)
    $j \leftarrow$ FOLLOW-BACKPOINTERS(b_k^j)
    return (i, j)
  return FOLLOW-BACKPOINTERS($k = root$) ▷ Backtrack to get the maximal tree.

2.3 DIORA CKY PARSING

To obtain a parse with DIORA, we populate an inside and outside chart using the input sentence. Then, we can extract the most likely parse based on our single grammar rule using the CKY procedure (Kasami, 1966; Younger, 1967).

It's true that using CKY produces the most likely parse given a set of grammar rules, although in the case of DIORA, the single grammar rule is only a weak abstraction for a PCFG. For this reason, including context during CKY might inform our parser to make different decision. We include context by adding the scalar value of the outside cell to each inside cell.

3 EXPERIMENTS

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

Our main baseline is the current state-of-the-art unsupervised parser PRPN(Shen et al., 2018). We compare our model against two size variants of this model which were used in Htut et al. (2018). Comparison of the number of parameters and maximum training sentence length are shown in 1.

Model	Word Dim	# Parameters	Max Length
DIORA	300	1,502,400	20
PRPN-UP	200	3,624,202	35
PRPN-LM	800	35,593,202	35

Table 1: Model Dimensions.

3.1 UNSUPERVISED PARSING

We first evaluate how well our model is able to predict a full unlabeled syntactic parse. We look at two data sets which have been used in prior work (Htut et al., 2018), The Wall Street Journal(WSJ) section of Penn Tree Bank (Marcus et al., 1994), and the automatic parses from MultiNLI (Williams et al., 2018b). WSJ has gold human annotated parses and MultiNLI contains automatic parses derived from the Stanford CoreNLP parser (Manning et al., 2014).

[1] https://github.com/anonymous/submission

We compare our model to left/right branching and balanced trees which are deterministically constructed. RL-SPINN (Yogatama et al., 2016) and ST-Gumbel (Choi et al., 2018) are chart parsing models trained to predict the downstream task of NLI.

3.1.1 RESULTS AND DISCUSSION

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 (+PP in Table 2). We see that PRPN-UP and DIORA benefit much more than PRPN-LM from this heuristic. This is consistent with qualitative analysis showing that DIORA and PRPN-UP incorrectly attach trailing punctuation much more than PRPN-LM. This heuristic simply attaches trailing punctuation to the root of the tree, regardless of its predicted attachment. We find this to be extremely effective, increasing our state-of-the-art WSJ parsing results by by over 3 absolute F1 points.

On the MultiNLI dataset, PRPN-LM is the top performing model without using the PP heuristic and DIORA outperforms PRPN-UP. Afterwards, PRPN-UP surpasses DIORA. However, it is worth noting that this is not actually a gold standard evaluation and instead evaluates the ability to replicate the output of a trained parser Manning et al. (2014).

	MultiNLI		**WSJ**	
Model	F1	Depth	F1	Depth
Left Branching	-	-	13.1	12.4
Right Branching	-	-	16.5	12.4
Random	27.0	4.4	21.4	5.3
Balanced	21.3	3.9	21.3	4.6
RL-SPINN†	18.8	8.6	13.2	-
- w/o Leaf GRU	18.1	8.6	13.2	-
ST-Gumbel†	23.7	4.1	20.1	-
- w/o Leaf GRU	27.5	4.6	25.0	-
PRPN-UP	48.4	4.9	40.6	5.9
PRPN-LM	50.2	5.0	43.5	6.2
DIORA	49.0	6.2	**43.8**	8.1
PRPN-UP +PP	54.6	4.9	46.1	5.9
PRPN-LM +PP	50.1	5.0	43.0	6.2
DIORA +PP	53.7	6.1	**46.9**	7.9

Table 2: Unsupervised Parsing. † indicates trained to optimize NLI task.We use the max unlabeled binary F1 across runs for PRPN-UP [2], PRPN-LM, and DIORA. F1 was calculated using the parse trees provided by Htut et al. (2018) and all results in the upper portion of the table were copied from Htut et al. (2018). +PP refers to post-processing heuristic to remove trailing punctuation explained in Section 3.1.

3.2 UNSUPERVISED PHRASE SEGMENTATION

In many scenarios, rather than a full parse, one is only concerned with extracting particular constituent phrases, such as entities, to be used for downstream analysis. In order 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. We calculate recall as the percentage of labeled constituents that appear in our predicted tree relative the total number of constituents in the gold tree. We separate these scores by type which are presented in Table 3.

3.2.1 RESULTS AND DISCUSSION

In Table 2 we see the breakdown of constituent recall across the 10 most common types. We see that PRPN-UP has the highest recall for the most common type noun-phrase, but drops in every other category. DIORA achieves the highest recall across the most types and is the only model to perform effectively on verb-phrases. However, DIORA performs poorly relative to PRPN at prepositional phrases.

Label	Count	DIORA	PRPN-UP	PRPN-LM
NP	297,687	0.620	**0.687**	0.597
VP	168,603	**0.569**	0.397	0.316
PP	116,338	0.338	0.499	**0.602**
S	87,714	**0.711**	0.629	0.625
SBAR	24,743	0.490	0.412	**0.554**
ADJP	12,261	**0.495**	0.343	0.360
QP	11,441	**0.624**	0.336	0.545
ADVP	5,812	0.437	0.392	**0.499**
PRN	2,971	**0.185**	0.108	0.138
SINV	2,563	**0.923**	0.889	0.905

Table 3: Segment recall from WSJ seperated by phrase type. The 10 most frequent phrase types are shown. Highest value in each row is bolded.

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 is a shallow parsing dataset containing spans of noun phrases, verb phrases, etc. CoNLL 2012 is a named entity dataset containing 19 different entity types.

For each of the labeled spans (greater than length 1) in the datasets, we generate a phrase representation and similarities are based on cosine distance. We report three numerical evaluations for both datasets precision@K, mean average precision (MAP), and dendogram purity (DP). We run a hierarchical clustering algorithm over the representations and computeDP (Kobren et al., 2017). Given any two points with the same gold label, the clustering tree is cut to form the minimal cluster containing both points. DP then calculates how pure that cluster is.

The first baseline we compare against produces phrase representations from averaging Glove vectors of the individual tokens within the span. The second uses ELMo (Peters et al., 2018a), a method for obtaining powerful, context dependent word embeddings that has led to many recent state-of-the-art results in NLP. We obtain phrases following the procedure described in (Peters et al., 2018b) and represent phrases as a function of its first and last hidden state. We look at two variants of ELMo[3]. ELMo-L1 produces token hidden states by only taking the bottom LSTM layer outputs, ELMo-Avg takes a flat average over all of the LSTM hidden state layers[4].

3.3.1 RESULTS

On the CoNLL 2000 dataset, we find that our model outperforms Glove and is competitive with ELMo. For CoNLL 2012, an named entity dataset, we find Glove to actually be the top performer

[3]we use the publically available code and pretrained mdoel from `https://allennlp.org/elmo`

[4]Note that we do not run this evaluation for PRPN because the authors released model parameters and complete parses but not a complete version of the code in order to run on new data.

under some metrics while our model is far behind. These results indicate that DIORA is capturing syntax quite well, but is currently missing semantics.

		CoNLL 2000				CoNLL 2012			
Model	Dim	DP	P@10	P@100	MAP	DP	P@10	P@100	MAP
Random	300	0.425	0.41	0.40	0.34	0.16	0.15	0.14	0.13
Glove	300	0.521	0.89	0.74	0.53	**0.392**	0.82	0.65	**0.47**
ELMo-L1	4096	0.632	0.96	0.83	0.55	0.352	0.85	0.68	0.39
- L2	4096	0.545	0.94	0.79	0.57	0.301	0.83	0.67	0.39
- Concat	8192	0.576	0.95	0.78	0.58	0.308	0.84	0.68	0.40
- Avg	4096	0.618	**0.97**	**0.84**	**0.67**	0.369	**0.87**	**0.73**	0.46
DIORA-In	200	0.620	0.89	0.77	0.61	0.303	0.77	0.54	0.36
- Out	200	0.581	0.69	0.65	0.47	0.200	0.35	0.24	0.18
- In/Out	400	0.615	0.89	0.79	0.62	0.308	0.76	0.54	0.35
- In/Out Ext.	400	**0.633**	0.89	0.79	0.62	0.311	0.77	0.54	0.35

Table 4: Dendogram purity, P@10, P@100, and MAP for labeled chunks from CoNLL-2000 and CoNLL 2012 datasets. For both metrics, higher is better. The top value in each column is bolded, or italicized if it is better than our model.

3.4 Qualitative Results

We show example trees from PRPN-LM and DIORA in 3.

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 latent tree was shown in Htut et al. (2018), which used a language modeling objective. The model in Liue et al. (2018) uses an inside chart and an outside procedure to calculate marginal probabilities use to align spans between sentences in entailment.

Neural Inside-Outside Parsers The Inside-Outside Recursive Neural Network (IORNN) in Le and Zuidema (2014) is closest to ours and is a graph-based dependency parser that produces a k-best list of parses, in contrast, DIORA produces the most likely parse given the learned the potential functions 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 Liue et al. (2018), has a single grammar rule that applies to any pair of constituents and does not use structural supervision.

Unsupervised Parsing and Segmentation Unsupervised segmentation (also called chunking) from raw text dates back to Ponvert et al. (2011). Another paper by the same authors (Ponvert et al., 2010) only looked at parsing certain low-level constituents. Earlier grammar induction models were evaluated against a subset of the WSJ treebank filtered to sentences of length 10 after removing punctuation (Klein and Manning, 2002; 2004) while DIORA is evaluated against two much larger datasets for unsupervised parsing, including the full WSJ treebank. Unsupervised segmentation with across parallel corpora was performed in Das and Petrov (2011). The source language had segment labels, the target language did not, but there are mapped translations between the two languages. Cohen et al. (2011) achieved unsupervised segmentation for parallel corpora without using mapped translations.

5 Conclusion

In this work we presented DIORA, a completely unsupervised method for inducing syntactic trees and segmentations over text. We showed that an auto encoder language modeling objective on top of inside-outside representations of latent tree chart parsers allows us to effectively learn syntactic

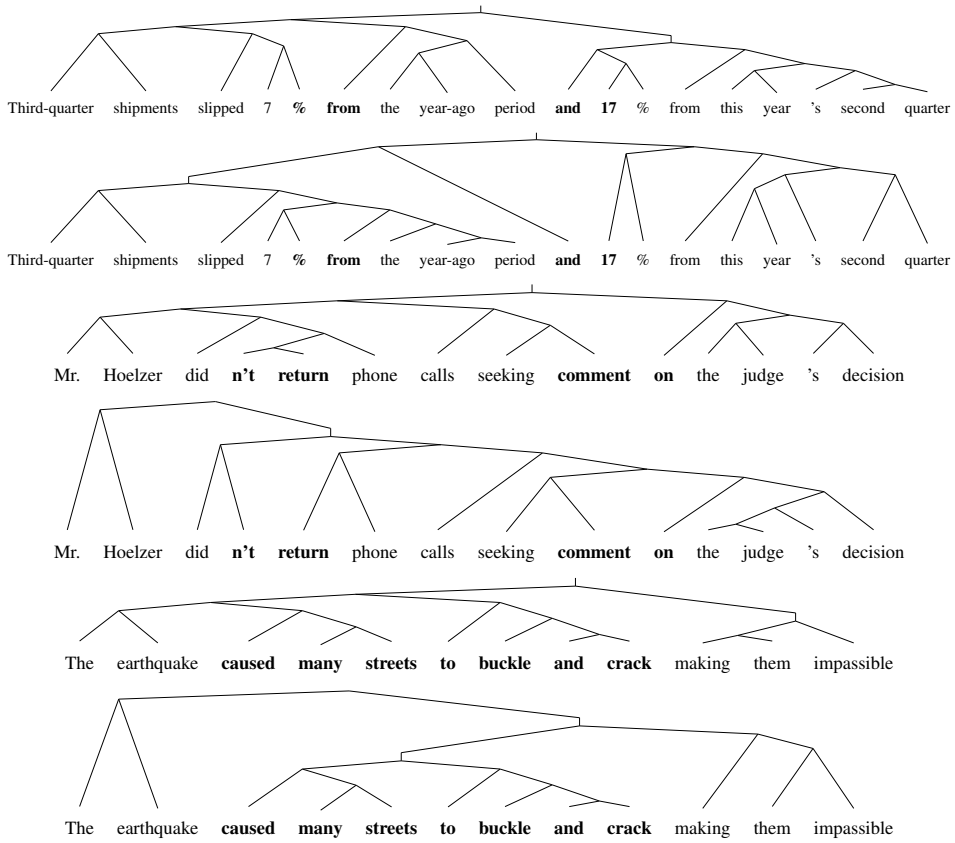


Figure 3: Pairs of example parses for the same sentence from two different models. For each pair, the top is the output of PRPN-LM and bottom was produced by DIORA. Bolden token pairs or spans indicate a parse error by PRPN that was correctly attached by DIORA. Some punctuation was removed for clarity of printed trees.

structure of language. In experiments on unsupervised parsing, chunking, and phrase representations we show our model is comparable to or outperforms current baselines, achieving the state-of-the-art performance on unsupervised parsing for the WSJ dataset. .

Future work can improve the current method by training larger models over much larger corpora including other domains and languages. While the current model seems to focus primarily on syntax, extra unsupervised objectives or light supervision could be injected into the learning procedure to encourage a more thorough capturing of semantics.

REFERENCES

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-1103`.

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

Roee Aharoni and Yoav Goldberg. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

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

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100*, 2016.

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

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. Learning to compose task-specific tree structures. In *In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

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

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

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

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. 2018.

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

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

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

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

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018b. URL `http://aclweb.org/anthology/N18-1101`.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 255–264, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098079. URL `http://doi.acm.org/10.1145/3097983.3098079`.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018a.

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

Yang Liue, Matt Gardner, and Mirella Lapata. Structured alignment networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium, October 2018.

Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 729–739, 2014.

Greg Durrett and Dan Klein. Neural crf parsing. In *ACL*, 2015.

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

Elias Ponvert, Jason Baldridge, and Katrin Erk. Simple unsupervised identification of low-level constituents. *2010 IEEE Fourth International Conference on Semantic Computing*, pages 24–31, 2010.

Dan Klein and Christopher D. Manning. A generative constituent-context model for improved grammar induction. In *ACL*, 2002.

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

Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL*, 2011.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

A APPENDIX

A.1 TRAINING DETAILS

All DIORA experiments are trained with these settings unless otherwise specified: we use the ALLNLI corpus including only sentences with length less than 20, stochastic gradient descent with a batch size of 256, and the model dimension set to 200. The input sentences are embedded using the 300D 480B GloVe embeddings (Pennington et al., 2014) and are not updated during training. Sentences are grouped into batches with uniform sentence length. Each cell in the chart has its L2-norm set to 1. Early stopping is done using the reconstruction objective evaluated on a held-out set. When depending on Noise Contrastive Estimation (as is the case in the reconstruction objective), we sample 3 negative examples per positive example.