Straight to the Tree: Constituency Parsing with Neural Syntactic Distance

Yikang Shen[*†]
MILA
University of Montréal

Zhouhan Lin[*†]
MILA
University of Montréal
AdeptMind Scholar

Athul Paul Jacob[†]
MILA
University of Waterloo

Alessandro Sordoni
Microsoft Research
Montréal, Canada

Aaron Courville and **Yoshua Bengio**
MILA
University of Montréal, CIFAR

Abstract

In this work, we propose a novel constituency parsing scheme. The model predicts a vector of real-valued scalars, named syntactic distances, for each split position in the input sentence. The syntactic distances specify the order in which the split points will be selected, recursively partitioning the input, in a top-down fashion. Compared to traditional shift-reduce parsing schemes, our approach is free from the potential problem of compounding errors, while being faster and easier to parallelize. Our model achieves competitive performance amongst single model, discriminative parsers in the PTB dataset and outperforms previous models in the CTB dataset.

1 Introduction

Devising fast and accurate constituency parsing algorithms is an important, long-standing problem in natural language processing. Parsing has been useful for incorporating linguistic prior in several related tasks, such as relation extraction, paraphrase detection (Callison-Burch, 2008), and more recently, natural language inference (Bowman et al., 2016) and machine translation (Eriguchi et al., 2017).

Neural network-based approaches relying on dense input representations have recently achieved competitive results for constituency parsing (Vinyals et al., 2015; Cross and Huang, 2016; Liu and Zhang, 2017b; Stern et al., 2017a). Generally speaking, either these approaches produce the parse tree sequentially, by governing the sequence of transitions in a transition-based parser (Nivre, 2004; Zhu et al., 2013; Chen and Manning, 2014; Cross and Huang, 2016), or use a chart-based approach by estimating non-linear potentials and performing exact structured inference by dynamic programming (Finkel et al., 2008; Durrett and Klein, 2015; Stern et al., 2017a).

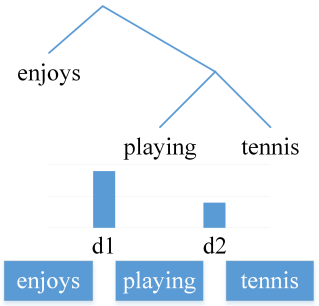


Figure 1: An example of how syntactic distances ($d1$ and $d2$) describe the structure of a parse tree: consecutive words with larger predicted distance are split earlier than those with smaller distances, in a process akin to divisive clustering.

Transition-based models decompose the structured prediction problem into a sequence of local decisions. This enables fast greedy decoding but also leads to compounding errors because the model is never exposed to its own mistakes during training (Daumé et al., 2009). Solutions to this problem usually complexify the training procedure by using structured training through beam-search (Weiss et al., 2015; Andor et al., 2016) and dynamic oracles (Goldberg and Nivre, 2012; Cross and Huang, 2016). On the other hand, chart-based models can incorporate structured loss functions during training and benefit from exact inference via the CYK algorithm but suffer from higher computational cost during decoding (Durrett and Klein, 2015; Stern et al., 2017a).

In this paper, we propose a novel, fully-parallel

[*] Equal contribution. Corresponding authors: yikang.shen@umontreal.ca, zhouhan.lin@umontreal.ca.
[†] Work done while at Microsoft Research, Montreal.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 1171–1180
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

model for constituency parsing, based on the concept of "syntactic distance", recently introduced by (Shen et al., 2017) for language modeling. To construct a parse tree from a sentence, one can proceed in a top-down manner, recursively splitting larger constituents into smaller constituents, where the order of the splits defines the hierarchical structure. The syntactic distances are defined for each possible split point in the sentence. The order induced by the syntactic distances fully specifies the order in which the sentence needs to be recursively split into smaller constituents (Figure 1): in case of a binary tree, there exists a one-to-one correspondence between the ordering and the tree. Therefore, our model is trained to reproduce the ordering between split points induced by the ground-truth distances by means of a margin rank loss (Weston et al., 2011). Crucially, our model works *in parallel*: the estimated distance for each split point is produced independently from the others, which allows for an easy parallelization in modern parallel computing architectures for deep learning, such as GPUs. Along with the distances, we also train the model to produce the constituent labels, which are used to build the fully labeled tree.

Our model is fully parallel and thus does not require computationally expensive structured inference during training. Mapping from syntactic distances to a tree can be efficiently done in $\mathcal{O}(n \log n)$, which makes the decoding computationally attractive. Despite our strong conditional independence assumption on the output predictions, we achieve good performance for single model discriminative parsing in PTB (91.8 F1) and CTB (86.5 F1) matching, and sometimes outperforming, recent chart-based and transition-based parsing models.

2 Syntactic Distances of a Parse Tree

In this section, we start from the concept of syntactic distance introduced in Shen et al. (2017) for unsupervised parsing via language modeling and we extend it to the supervised setting. We propose two algorithms, one to convert a parse tree into a compact representation based on distances between consecutive words, and another to map the inferred representation back to a complete parse tree. The representation will later be used for supervised training. We formally define the syntactic distances of a parse tree as follows:

Algorithm 1 Binary Parse Tree to Distance

($\cup$ represents the concatenation operator of lists)

function DISTANCE(node)
  if node **is** leaf **then**
    $\mathbf{d} \leftarrow []$
    $\mathbf{c} \leftarrow []$
    $\mathbf{t} \leftarrow [\text{node.tag}]$
    $h \leftarrow 0$
  else
    $\text{child}_l, \text{child}_r \leftarrow$ children of node
    $\mathbf{d}_l, \mathbf{c}_l, \mathbf{t}_l, h_l \leftarrow \text{Distance}(\text{child}_l)$
    $\mathbf{d}_r, \mathbf{c}_r, \mathbf{t}_r, h_r \leftarrow \text{Distance}(\text{child}_r)$
    $h \leftarrow \max(h_l, h_r) + 1$
    $\mathbf{d} \leftarrow \mathbf{d}_l \cup [h] \cup \mathbf{d}_r$
    $\mathbf{c} \leftarrow \mathbf{c}_l \cup [\text{node.label}] \cup \mathbf{c}_r$
    $\mathbf{t} \leftarrow \mathbf{t}_l \cup \mathbf{t}_r$
  end if
  return $\mathbf{d}, \mathbf{c}, \mathbf{t}, h$
end function

Definition 2.1. Let $\mathbf{T}$ be a parse tree that contains a set of leaves $(w_0, ..., w_n)$. The height of the lowest common ancestor for two leaves (w_i, w_j) is noted as $\tilde{d}_j^i$. The syntactic distances of $\mathbf{T}$ can be any vector of scalars $\mathbf{d} = (d_1, ..., d_n)$ that satisfy:

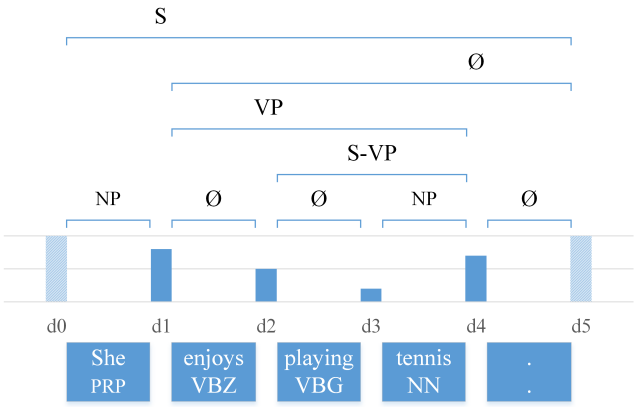
$$\text{sign}(d_i - d_j) = \text{sign}(\tilde{d}_i^{i-1} - \tilde{d}_j^{j-1}) \qquad (1)$$

In other words, $\mathbf{d}$ induces the same ranking order as the quantities $\tilde{d}_i^j$ computed between pairs of consecutive words in the sequence, i.e. $(\tilde{d}_1^0, ..., \tilde{d}_n^{n-1})$. Note that there are $n-1$ syntactic distances for a sentence of length n.

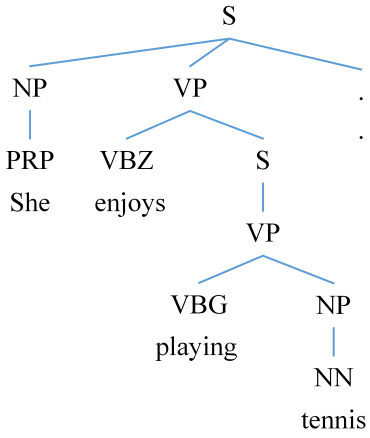
Example 2.1. Consider the tree in Fig. 1 for which $\tilde{d}_1^0 = 2$, $\tilde{d}_2^1 = 1$. An example of valid syntactic distances for this tree is any $\mathbf{d} = (d_1, d_2)$ such that $d_1 > d_2$.

Given this definition, the parsing model predicts a sequence of scalars, which is a more natural setting for models based on neural networks, rather than predicting a set of spans. For comparison, in most of the current neural parsing methods, the model needs to output a sequence of transitions (Cross and Huang, 2016; Chen and Manning, 2014).

Let us first consider the case of a binary parse tree. Algorithm 1 provides a way to convert it to a tuple $(\mathbf{d}, \mathbf{c}, \mathbf{t})$, where $\mathbf{d}$ contains the height of the inner nodes in the tree following a left-to-right (in order) traversal, $\mathbf{c}$ the constituent labels for each node in the same order and $\mathbf{t}$ the part-of-speech



(a) Boxes in the bottom are words and their corresponding POS tags predicted by an external tagger. The vertical bars in the middle are the syntactic distances, and the brackets on top of them are labels of constituents. The bottom brackets are the predicted unary label for each words, and the upper brackets are predicted labels for other constituent.



(b) The corresponding inferred grammar tree.

Figure 2: Inferring the parse tree with Algorithm 2 given distances, constituent labels, and POS tags. Starting with the full sentence, we pick split point 1 (as it is assigned to the larger distance) and assign label S to span (0,5). The left child span (0,1) is assigned with a tag PRP and a label NP, which produces an unary node and a terminal node. The right child span (1,5) is assigned the label $\varnothing$, coming from implicit binarization, which indicates that the span is not a real constituent and all of its children are instead direct children of its parent. For the span (1,5), the split point 4 is selected. The recursion of splitting and labeling continues until the process reaches a terminal node.

Algorithm 2 Distance to Binary Parse Tree

function TREE($\mathbf{d}$,$\mathbf{c}$,$\mathbf{t}$)
  if $\mathbf{d} = []$ **then**
    node $\leftarrow$ Leaf($\mathbf{t}$)
  else
    $i \leftarrow \arg\max_i(\mathbf{d})$
    $\text{child}_l \leftarrow \text{Tree}(\mathbf{d}_{<i}, \mathbf{c}_{<i}, \mathbf{t}_{<i})$
    $\text{child}_r \leftarrow \text{Tree}(\mathbf{d}_{>i}, \mathbf{c}_{>i}, \mathbf{t}_{\geq i})$
    $\text{node} \leftarrow \text{Node}(\text{child}_l, \text{child}_r, \mathbf{c}_i)$
  end if
  return node
end function

(POS) tags of each word in the left-to-right order. $\mathbf{d}$ is a valid vector of syntactic distances satisfying Definition 2.1.

Once a model has learned to predict these variables, Algorithm 2 can reconstruct a unique binary tree from the output of the model $(\hat{\mathbf{d}}, \hat{\mathbf{c}}, \hat{\mathbf{t}})$. The idea in Algorithm 2 is similar to the top-down parsing method proposed by Stern et al. (2017a), but differs in one key aspect: at each recursive call, there is no need to estimate the confidence for every split point. The algorithm simply chooses the split point i with the maximum $\hat{d}_i$, and assigns to the span the predicted label $\hat{c}_i$. This makes the running time of our algorithm to be in $\mathcal{O}(n \log n)$, compared to the $\mathcal{O}(n^2)$ of the greedy top-down algorithm by (Stern et al., 2017a). Figure 2 shows an example of the reconstruction of parse tree. Alternatively, the tree reconstruction process can also be done in a bottom-up manner, which requires the recursive composition of adjacent spans according to the ranking induced by their syntactic distance, a process akin to agglomerative clustering.

One potential issue is the existence of unary and n-ary nodes. We follow the method proposed by Stern et al. (2017a) and add a special empty label $\varnothing$ to spans that are not themselves full constituents but simply arise during the course of implicit binarization. For the unary nodes that contains one nonterminal node, we take the common approach of treating these as additional atomic labels alongside all elementary nonterminals (Stern et al., 2017a). For all terminal nodes, we determine whether it belongs to a unary chain or not by predicting an additional label. If it is predicted with a label different from the empty label, we conclude that it is a direct child of a unary constituent with that label. Otherwise if it is predicted to have an empty label, we conclude that it is a child of a bigger constituent which has other constituents or words as its siblings.

An n-ary node can arbitrarily be split into binary nodes. We choose to use the leftmost split point. The split point may also be chosen based on model prediction during training. Recovering an n-ary parse tree from the predicted binary tree simply requires removing the empty nodes and split combined labels corresponding to unary chains.

Algorithm 2 is a divide-and-conquer algorithm. The running time of this procedure is $\mathcal{O}(n \log n)$. However, the algorithm is naturally adapted for execution in a parallel environment, which can further reduce its running time to $\mathcal{O}(\log n)$.

3 Learning Syntactic Distances

We use neural networks to estimate the vector of syntactic distances for a given sentence. We use a modified hinge loss, where the target distances are generated by the tree-to-distance conversion given by Algorithm 1. Section 3.1 will describe in detail the model architecture, and Section 3.2 describes the loss we use in this setting.

3.1 Model Architecture

Given input words $\mathbf{w} = (w_0, w_1, ..., w_n)$, we predict the tuple $(\mathbf{d}, \mathbf{c}, \mathbf{t})$. The POS tags $\mathbf{t}$ are given by an external Part-Of-Speech (POS) tagger. The syntactic distances $\mathbf{d}$ and constituent labels $\mathbf{c}$ are predicted using a neural network architecture that stacks recurrent (LSTM (Hochreiter and Schmidhuber, 1997)) and convolutional layers.

Words and tags are first mapped to sequences of embeddings $\mathbf{e}^{\mathrm{w}}_0, ..., \mathbf{e}^{\mathrm{w}}_n$ and $\mathbf{e}^{\mathrm{t}}_0, ..., \mathbf{e}^{\mathrm{t}}_n$. Then the word embeddings and the tag embeddings are concatenated together as inputs for a stack of bidirectional LSTM layers:

$$\mathbf{h}^{\mathrm{w}}_0, ..., \mathbf{h}^{\mathrm{w}}_n = \mathrm{BiLSTM}_{\mathrm{w}}([\mathbf{e}^{\mathrm{w}}_0, \mathbf{e}^{\mathrm{t}}_0], ..., [\mathbf{e}^{\mathrm{w}}_n, \mathbf{e}^{\mathrm{t}}_n]) \tag{2}$$

where $\mathrm{BiLSTM}_{\mathrm{w}}(\cdot)$ is the word-level bidirectional layer, which gives the model enough capacity to capture long-term syntactical relations between words.

To predict the constituent labels for each word, we pass the hidden states representations $\mathbf{h}^{\mathrm{w}}_0, ..., \mathbf{h}^{\mathrm{w}}_n$ through a 2-layer network $\mathrm{FF}^{\mathrm{w}}_c$, with softmax output:

$$p(c^{\mathbf{w}}_i | \mathbf{w}) = \mathrm{softmax}(\mathrm{FF}^{\mathrm{w}}_c(\mathbf{h}^{\mathrm{w}}_i)) \tag{3}$$

To compose the necessary information for inferring the syntactic distances and the constituency label information, we perform an additional convolution:

$$\mathbf{g}^{\mathrm{s}}_1, \ldots, \mathbf{g}^{\mathrm{s}}_n = \mathrm{CONV}(\mathbf{h}^{\mathrm{w}}_0, ..., \mathbf{h}^{\mathrm{w}}_n) \tag{4}$$

where $\mathbf{g}^{\mathrm{s}}_i$ can be seen as a draft representation for each split position in Algorithm 2. Note that the subscripts of g^s_is start with 1, since we have $n-1$ positions as non-terminal constituents. Then, we stack a bidirectional LSTM layer on top of $\mathbf{g}^{\mathrm{s}}_i$:

$$\mathbf{h}^{\mathrm{s}}_1, ..., \mathbf{h}^{\mathrm{s}}_n = \mathrm{BiLSTM}_{\mathrm{s}}(\mathbf{g}^{\mathrm{s}}_1, \ldots, \mathbf{g}^{\mathrm{s}}_n) \tag{5}$$

where $\mathrm{BiLSTM}_{\mathrm{s}}$ fine-tunes the representation by conditioning on other split position representations. Interleaving between LSTM and convolution layers turned out empirically to be the best choice over multiple variations of the model, including using self-attention (Vaswani et al., 2017) instead of LSTM.

To calculate the syntactic distances for each position, the vectors $\mathbf{h}^{\mathrm{s}}_1, \ldots, \mathbf{h}^{\mathrm{s}}_n$ are transformed through a 2-layer feed-forward network FF_d with a single output unit (this can be done in parallel with 1x1 convolutions), with no activation function at the output layer:

$$\hat{d}_i = \mathrm{FF}_d(\mathbf{h}^s_i), \tag{6}$$

For predicting the constituent labels, we pass the same representations $\mathbf{h}^{\mathrm{s}}_1, \ldots, \mathbf{h}^{\mathrm{s}}_n$ through another 2-layer network $\mathrm{FF}^{\mathrm{s}}_c$, with softmax output.

$$p(c^{\mathrm{s}}_i | \mathbf{w}) = \mathrm{softmax}(\mathrm{FF}^{\mathrm{s}}_c(\mathbf{h}^{\mathrm{s}}_i)) \tag{7}$$

The overall architecture is shown in Figure 2a. Since the output $(\mathbf{d}, \mathbf{c}, \mathbf{t})$ can be unambiguously transfered to a unique parse tree, the model implicitly makes all parsing decisions inside the recurrent and convolutional layers.

3.2 Objective

Given a set of training examples $\mathcal{D} = \{\langle \mathbf{d}_k, \mathbf{c}_k, \mathbf{t}_k, \mathbf{w}_k \rangle\}^K_{k=1}$, the training objective is the sum of the prediction losses of syntactic distances $\mathbf{d}_k$ and constituent labels $\mathbf{c}_k$.

Due to the categorical nature of variable $\mathbf{c}$, we use a standard softmax classifier with a cross-entropy loss L_{label} for constituent labels, using the estimated probabilities obtained in Eq. 3 and 7.

A naïve loss function for estimating syntactic distances is the mean-squared error (MSE):

$$L^{\text{mse}}_{\text{dist}} = \sum_i (d_i - \hat{d}_i)^2 \tag{8}$$

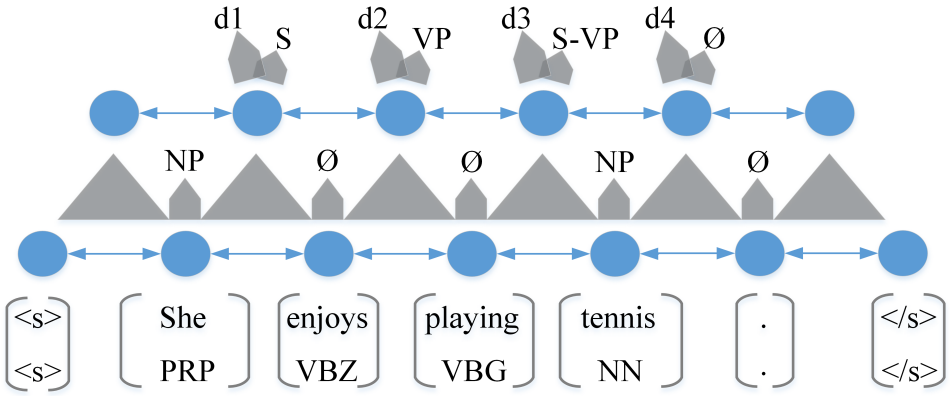


Figure 3: The overall visualization of our model. Circles represent hidden states, triangles represent convolution layers, block arrows represent feed-forward layers, arrows represent recurrent connections. The bottom part of the model predicts unary labels for each input word. The $\emptyset$ is treated as a special label together with other labels. The top part of the model predicts the syntactic distances and the constituent labels. The inputs of model are the word embeddings concatenated with the POS tag embeddings. The tags are given by an external Part-Of-Speech tagger.

The MSE loss forces the model to regress on the exact value of the true distances. Given that only the *ranking* induced by the ground-truth distances in $\mathbf{d}$ is important, as opposed to the absolute values themselves, using an MSE loss over-penalizes the model by ignoring ranking equivalence between different predictions.

Therefore, we propose to minimize a pair-wise learning-to-rank loss, similar to those proposed in (Burges et al., 2005). We define our loss as a variant of the hinge loss as:

$$L_{\text{dist}}^{\text{rank}} = \sum_{i,j>i} [1 - \text{sign}(d_i - d_j)(\hat{d}_i - \hat{d}_j)]^+, \quad (9)$$

where $[x]^+$ is defined as $\max(0, x)$. This loss encourages the model to reproduce the full ranking order induced by the ground-truth distances. The final loss for the overall model is just the sum of individual losses $L = L_{\text{label}} + L_{\text{dist}}^{\text{rank}}$.

4 Experiments

We evaluate our model described above on 2 different datasets, the standard Wall Street Journal (WSJ) part of the Penn Treebank (PTB) dataset, and the Chinese Treebank (CTB) dataset.

For evaluating the F1 score, we use the standard `evalb`[1] tool. We provide both labeled and unlabeled F1 score, where the former takes into consideration the constituent label for each predicted constituent, while the latter only considers the position of the constituents. In the tables below, we report the labeled F1 scores for comparison with previous work, as this is the standard metric usually reported in the relevant literature.

4.1 Penn Treebank

For the PTB experiments, we follow the standard train/valid/test separation and use sections 2-21 for training, section 22 for development and section 23 for test set. Following this split, the dataset has 45K training sentences and 1700, 2416 sentences for valid/test respectively. The placeholders with the `-NONE-` tag are stripped from the dataset during preprocessing. The POS tags are predicted with the Stanford Tagger (Toutanova et al., 2003).

We use a hidden size of 1200 for each direction on all LSTMs, with 0.3 dropout in all the feed-forward connections, and 0.2 recurrent connection dropout (Merity et al., 2017). The convolutional filter size is 2. The number of convolutional channels is 1200. As a common practice for neural network based NLP models, the embedding layer that maps word indexes to word embeddings is randomly initialized. The word embeddings are sized 400. Following (Merity et al., 2017), we randomly swap an input word embedding during training with the zero vector with probability of 0.1. We found this helped the model to generalize better. Training is conducted with Adam algorithm with l2 regularization decay 1×10^{-6}. We pick the result obtaining the highest labeled F1

[1] http://nlp.cs.nyu.edu/evalb/

Model	LP	LR	F1
Single Model			
Vinyals et al. (2015)	-	-	88.3
Zhu et al. (2013)	90.7	90.2	90.4
Dyer et al. (2016)	-	-	89.8
Watanabe and Sumita (2015)	-	-	90.7
Cross and Huang (2016)	92.1	90.5	91.3
Liu and Zhang (2017b)	92.1	91.3	91.7
Stern et al. (2017a)	93.2	90.3	91.8
Liu and Zhang (2017a)	-	-	91.8
Gaddy et al. (2018)	-	-	92.1
Stern et al. (2017b)	92.5	92.5	92.5
Our Model	92.0	91.7	91.8
Ensemble			
Shindo et al. (2012)	-	-	92.4
Vinyals et al. (2015)	-	-	90.5
Semi-supervised			
Zhu et al. (2013)	91.5	91.1	91.3
Vinyals et al. (2015)	-	-	92.8
Re-ranking			
Charniak and Johnson (2005)	91.8	91.2	91.5
Huang (2008)	91.2	92.2	91.7
Dyer et al. (2016)	-	-	93.3

Table 1: Results on the PTB dataset WSJ test set, Section 23. LP, LR represents labeled precision and recall respectively.

on the validation set, and report the corresponding test F1, together with other statistics. We report our results in Table 1. Our best model obtains a labeled F1 score of 91.8 on the test set (Table 1). Detailed dev/test set performances, including label accuracy is reported in Table 3.

Our model performs achieves good performance for single-model constituency parsing trained without external data. The best result from (Stern et al., 2017b) is obtained by a generative model. Very recently, we came to knowledge of Gaddy et al. (2018), which uses character-level LSTM features coupled with chart-based parsing to improve performance. Similar sub-word features can be also used in our model. We leave this investigation for future works. For comparison, other models obtaining better scores either use ensembles, benefit from semi-supervised learning, or recur to re-ranking of a set of candidates.

4.2 Chinese Treebank

We use the Chinese Treebank 5.1 dataset, with articles 001-270 and 440-1151 for training, articles

Model	LP	LR	F1
Single Model			
Charniak (2000)	82.1	79.6	80.8
Zhu et al. (2013)	84.3	82.1	83.2
Wang et al. (2015)	-	-	83.2
Watanabe and Sumita (2015)	-	-	84.3
Dyer et al. (2016)	-	-	84.6
Liu and Zhang (2017b)	85.9	85.2	85.5
Liu and Zhang (2017a)	-	-	86.1
Our Model	86.6	86.4	86.5
Semi-supervised			
Zhu et al. (2013)	86.8	84.4	85.6
Wang and Xue (2014)	-	-	86.3
Wang et al. (2015)	-	-	86.6
Re-ranking			
Charniak and Johnson (2005)	83.8	80.8	82.3
Dyer et al. (2016)	-	-	86.9

Table 2: Test set performance comparison on the CTB dataset

301-325 as development set, and articles 271-300 for test set. This is a standard split in the literature (Liu and Zhang, 2017b). The `-NONE-` tags are stripped as well. The hidden size for the LSTM networks is set to 1200. We use a dropout rate of 0.4 on the feed-forward connections, and 0.1 recurrent connection dropout. The convolutional layer has 1200 channels, with a filter size of 2. We use 400 dimensional word embeddings. During training, input word embeddings are randomly swapped with the zero vector with probability of 0.1. We also apply a l2 regularization weighted by 1×10^{-6} on the parameters of the network. Table 2 reports our results compared to other benchmarks. To the best of our knowledge, we set a new state-of-the-art for single-model parsing achieving 86.5 F1 on the test set. The detailed statistics are shown in Table 3.

4.3 Ablation Study

We perform an ablation study by removing components from a network trained with the best set of hyperparameters, and re-train the ablated version from scratch. This gives an idea of the relative contributions of each of the components in the model. Results are reported in Table 4. It seems that the top LSTM layer has a relatively big impact on performance. This may give additional capacity to the model for capturing long-term dependencies useful for label prediction. We also exper-

dev/test result		Prec.	Recall	F1	label accuracy
PTB	labeled	91.7/92.0	91.8/91.7	91.8/91.8	94.9/95.4%
	unlabeled	93.0/93.2	93.0/92.8	93.0/93.0	
CTB	labeled	89.4/86.6	89.4/86.4	89.4/86.5	92.2/91.1%
	unlabeled	91.1/88.9	91.1/88.6	91.1/88.8	

Table 3: Detailed experimental results on PTB and CTB datasets

Model	LP	LR	F1
Full model	92.0	91.7	91.8
w/o top LSTM	91.0	90.5	90.7
w. embedding	91.9	91.6	91.7
w. MSE loss	90.3	90.0	90.1

Table 4: Ablation test on the PTB dataset. "w/o top LSTM" is the full model without the top LSTM layer. "w. embedding" stands for the full model using the pretrained word embeddings. "w. MSE loss" stands for the full model trained with MSE loss.

imented by using 300D GloVe (Pennington et al., 2014) embedding for the input layer but this didn't yield improvements over the model's best performance. Unsurprisingly, the model trained with MSE loss underperforms considerably a model trained with the rank loss.

4.4 Parsing Speed

The prediction of syntactic distances can be batched in modern GPU architectures. The distance to tree conversion is a $\mathcal{O}(n \log n)$ (n stand for the number of words in the input sentence) divide-and-conquer algorithm. We compare the parsing speed of our parser with other state-of-the-art neural parsers in Table 5. As the syntactic distance computation can be performed in parallel within a GPU, we first compute the distances in a batch, then we iteratively decode the tree with Algorithm 2. It is worth to note that this comparison may be unfair since some of the reported results may use very different hardware settings. We couldn't find the source code to re-run them on our hardware, to give a fair enough comparison. In our setting, we use an NVIDIA TITAN Xp graphics card for running the neural network part, and the distance to tree inference is run on an Intel Core i7-6850K CPU, with 3.60GHz clock speed.

Model	# sents/sec
Petrov and Klein (2007)	6.2
Zhu et al. (2013)	89.5
Liu and Zhang (2017b)	79.2
Stern et al. (2017a)	75.5
Our model	111.1
Our model w/o tree inference	351

Table 5: Parsing speed in sentences per second on the PTB dataset.

5 Related Work

Parsing natural language with neural network models has recently received growing attention. These models have attained state-of-the-art results for dependency parsing (Chen and Manning, 2014) and constituency parsing (Dyer et al., 2016; Cross and Huang, 2016; Coavoux and Crabbé, 2016). Early work in neural network based parsing directly use a feed-forward neural network to predict parse trees (Chen and Manning, 2014). Vinyals et al. (2015) use a sequence-to-sequence framework where the decoder outputs a linearized version of the parse tree given an input sentence. Generally, in these models, the correctness of the output tree is not strictly ensured (although empirically observed).

Other parsing methods ensure structural consistency by operating in a transition-based setting (Chen and Manning, 2014) by parsing either in the top-down direction (Dyer et al., 2016; Liu and Zhang, 2017b), bottom-up (Zhu et al., 2013; Watanabe and Sumita, 2015; Cross and Huang, 2016) and recently in-order (Liu and Zhang, 2017a). Transition-based methods generally suffer from compounding errors due to exposure bias: during testing, the model is exposed to a very different regime (i.e. decisions sampled from the model itself) than what was encountered during training (i.e. the ground-truth decisions) (Daumé et al., 2009; Goldberg and Nivre, 2012). This can have catastrophic effects on test performance but

can be mitigated to a certain extent by using beam-search instead of greedy decoding. (Stern et al., 2017b) proposes an effective inference method for generative parsing, which enables direct decoding in those models. More complex training methods have been devised in order to alleviate this problem (Goldberg and Nivre, 2012; Cross and Huang, 2016). Other efforts have been put into neural chart-based parsing (Durrett and Klein, 2015; Stern et al., 2017a) which ensure structural consistency and offer exact inference with CYK algorithm. (Gaddy et al., 2018) includes a simplified CYK-style inference, but the complexity still remains in $O(n^3)$.

In this work, our model learns to produce a particular representation of a tree in parallel. Representations can be computed in parallel, and the conversion from representation to a full tree can efficiently be done with a divide-and-conquer algorithm. As our model outputs decisions in parallel, our model doesn't suffer from the exposure bias. Interestingly, a series of recent works, both in machine translation (Gu et al., 2018) and speech synthesis (Oord et al., 2017), considered the sequence of output variables conditionally independent given the inputs.

6 Conclusion

We presented a novel constituency parsing scheme based on predicting real-valued scalars, named syntactic distances, whose ordering identify the sequence of top-down split decisions. We employ a neural network model that predicts the distances $\mathbf{d}$ and the constituent labels $\mathbf{c}$. Given the algorithms presented in Section 2, we can build an unambiguous mapping between each $(\mathbf{d}, \mathbf{c}, \mathbf{t})$ and a parse tree. One peculiar aspect of our model is that it predicts split decisions *in parallel*. Our experiments show that our model can achieve strong performance compare to previous models, while being significantly more efficient. Since the architecture of model is no more than a stack of standard recurrent and convolution layers, which are essential components in most academic and industrial deep learning frameworks, the deployment of this method would be straightforward.

Acknowledgement

The authors would like to thank Compute Canada for providing the computational resources. The authors would also like to thank Jackie Chi Kit Cheung for the helpful discussions. Zhouhan Lin would like to thank AdeptMind for generously supporting his research via scholarship.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2442–2452.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1466–1477.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*. pages 89–96.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 196–205.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, pages 132–139.

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

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 740–750.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 172–182.

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*. Association for Computational Linguistics, pages 1–âĂŞ11.

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

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 302–312.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 199âĂŞ–209.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 72–78.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*. Association for Computational Linguistics, pages 959–967.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. WhatâĂŹs going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*. pages 959–976.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proceedings of International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, pages 586–594.

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association of Computational Linguistics* 5(1):413–424.

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics* 5:45–58.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182* .

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. 2017. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433* .

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. pages 404–411.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. In *Proceedings of the International Conference on Learning Representations*.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, pages 440–448.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017a. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 818–827.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017b. Effective inference for generative neural parsing. *arXiv preprint arXiv:1707.08976* .

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. pages 6000–6010.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1138–1147.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 733–742.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Volume 1, Long Papers*. pages 1169–1179.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158* .

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. pages 2764–2770.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 434–443.