Cooperative Learning of Disjoint Syntax and Semantics

Serhii Havrylov *
ILCC, University of Edinburgh / Edinburgh, UK
s.havrylov@ed.ac.uk

Germán Kruszewski & Armand Joulin
Facebook AI Research
{germank,ajoulin}@fb.com

Abstract

There has been considerable attention devoted to models that learn to jointly infer an expression's syntactic structure and its semantics. Yet, Nangia and Bowman (2018) has recently shown that the current best systems fail to learn the correct parsing strategy on mathematical expressions generated from a simple context-free grammar. In this work, we present a recursive model inspired by Choi et al. (2018) that reaches near perfect accuracy on this task. Our model is composed of two separated modules for syntax and semantics. They are cooperatively trained with standard continuous and discrete optimization schemes. Our model does not require any linguistic structure for supervision and its recursive nature allows for out-of-domain generalization with little loss in performance. Additionally, our approach performs competitively on several natural language tasks, such as Natural Language Inference or Sentiment Analysis.

1 Introduction

Standard linguistic theories propose that natural language is structured as nested constituents organised in the form of a tree (Partee et al., 1990). However, most popular models, such as the Long Sort-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997), process text without imposing a grammatical structure. To bridge this gap between theory and practice models that process linguistic expressions in a tree-structured manner have been considered in recent work (Socher et al., 2013b; Tai et al., 2015; Zhu et al., 2015; Bowman et al., 2016). These tree-based models explicitly require access to the syntactic structure for the text, which is not entirely satisfactory.

Indeed, parse tree level supervision requires a significant amount of annotations from expert linguists. These trees have been annotated with different goals in mind than the tasks we are using them for. Such discrepancy may result in a deterioration of the performance of models relying on them. Recently, several attempts were made to learn these models without explicit supervision for the parser (Yogatama et al., 2016; Maillard et al., 2017; Choi et al., 2018). However, Williams et al. (2018a) has recently shown that the structures learned by these models cannot be ascribed to discovering meaningful syntactic structure. These models even fail to learn the simple context-free grammar of nested mathematical operations (Nangia and Bowman, 2018).

In this work, we present an extension of Choi et al. (2018), that successfully learns these simple grammars while preserving competitive performance on several standard linguistic tasks. Contrary to previous work, our model makes a clear distinction between the parser and the compositional function. These two modules are trained with different algorithms, cooperating to build a semantic representation that optimizes the objective function. The parser's goal is to generate a tree structure for the sentence. The compositional function follows this structure to produce the sentence representation. Our model contains a continuous component, the compositional function, and a discrete one, the parser. The whole system is trained end-to-end with a mix of reinforcement learning and gradient descent. Drozdov and Bowman (2017) has noticed the difficulty of mixing these two optimization schemes without one dominating the other. This typically leads to the "coadaptation problem" where the parser simply follows the compositional function and fails to produce meaningful syntactic structures. In this work, we show that this pitfall can be avoided

*Work done while the author was an intern at Facebook AI Research.

by synchronising the learning paces of the two optimisation schemes. This is achieved by combining several recent advances in reinforcement learning. First, we use input-dependent control variates to reduce the variance of our gradient estimates (Ross, 1997). Then, we apply multiple gradient steps to the parser's policy while controlling for its learning pace using the Proximal Policy Optimization (PPO) of Schulman et al. (2017). The code for our model is publicly available[1].

2 Preliminaries

In this section, we present existing works on Recursive Neural Networks and their training in the absence of supervision on the syntactic structures.

2.1 Recursive Neural Networks

A Recursive Neural Network (RvNN) has its architecture defined by a directed acyclic graph (DAG) given alongside with an input sequence (Goller and Kuchler, 1996). RvNNs are commonly used in NLP to generate sentence representation that leverages available syntactic information, such as a constituency or a dependency parse trees (Socher et al., 2011).

Given an input sequence and its associated DAG, a RvNN processes the sequence by applying a transformation to the representations of the tokens lying on the lowest levels of the DAG. This transformation, or compositional function, merges these representations into representations for the nodes on the next level of the DAG. This process is repeated recursively along the graph structure until the top-level nodes are reached. In this work, we assume that the compositional function is the same for every node in the graph.

Tree-LSTM. We focus on a specific type of RvNNs, the tree-based long short-term memory network (Tree-LSTM) of Tai et al. (2015) and Zhu et al. (2015). Its compositional function generalizes the LSTM cell of Hochreiter and Schmidhuber (1997) to tree-structured topologies, i.e.,

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{i} \\ \mathbf{f}_l \\ \mathbf{f}_r \\ \mathbf{o} \end{bmatrix} = \begin{bmatrix} \texttt{tanh} \\ \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{R} \begin{bmatrix} \mathbf{h}_l \\ \mathbf{h}_r \end{bmatrix} + \mathbf{b} \right),$$

$$\mathbf{c}_p = \mathbf{z} \odot \mathbf{i} + \mathbf{c}_l \odot \mathbf{f}_l + \mathbf{c}_r \odot \mathbf{f}_r,$$

$$\mathbf{h}_p = \texttt{tanh}(\mathbf{c}_p) \odot \mathbf{o},$$

where σ and $\texttt{tanh}$ are the sigmoid and hyperbolic tangent functions. Tree-LSTM cell is differentiable with respect to its recursion matrix $\mathbf{R}$, bias $\mathbf{b}$ and its input. The gradients of a Tree-LSTM can thus be computed with backpropagation through structure (BPTS) (Goller and Kuchler, 1996).

2.2 Learning with RvNNs

A tree-based RvNN is a function f_θ parameterized by a d dimensional vector θ that predicts an output y given an input x and a tree t. Given a dataset $\mathcal{D}$ of N triplets (x, t, y), the parameters of the RvNN are learned with the following minimisation problem:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{N} \sum_{(x,t,y) \in \mathcal{D}} \ell(f_\theta(x, t), y), \qquad (1)$$

where ℓ is a logistic regression function. These models need an externally provided parsing tree for each input sentence during both training and evaluation. Alternatives, such as the shift-reduce-based SPINN model of Bowman et al. (2016), learn an internal parser from the given trees. While these solutions do not need external trees during evaluation, they still require tree level annotations for training. More recent work has focused on learning a latent parser with no direct supervision.

2.3 Latent tree models

Latent tree models aim at jointly learning the compositional function f_θ and a parser without supervision on the syntactic structures (Yogatama et al., 2016; Maillard et al., 2017; Choi et al., 2018). The latent parser is defined as a parametric probability distribution over trees conditioned on the input sequence. The parameters of this tree distribution $p_\phi(.|x)$ are represented by a vector ϕ. Given a dataset $\mathcal{D}$ of pairs of input sequences x and outputs y, the parameters θ and ϕ are jointly learned by minimising the following objective function:

$$\min_{\theta,\phi} \mathcal{L}(\theta, \phi) = \frac{1}{N} \sum_{(x,y)} \mathbb{E}_\phi[\ell(f_\theta(x, t), y)], \qquad (2)$$

[1] http://github.com/anonymised_link

where $\mathbb{E}_\phi$ is the expectation following the $p_\phi(.|x)$ distribution. Learning a distribution over a set of discrete quantities involves a discrete optimisation scheme. For example, the RL-SPINN model of Yogatama et al. (2016) uses a mix of gradient descent for θ and REINFORCE for ϕ (Williams et al., 2018a). Drozdov and Bowman (2017) has recently observed that this optimisation strategy tends to produce poor parsers, e.g., parsers that only generate left-branching trees. The effect, called the coadaptation issue, is caused by both bias in the parsing strategy and a difference in convergence paces of continuous and discrete optimisers. Typically, the parameters θ are learned more rapidly than ϕ. This limits the exploration of the search space to parsing strategies similar to what was found at the beginning of the training.

2.3.1 Gumbel Tree-LSTM

In their Gumbel Tree-LSTM model, Choi et al. (2018) propose an alternative parsing strategy to avoid the coadaptation issue. Their parser incrementally merges a pair of consecutive constituents until a single one remains. This strategy reduces the bias towards certain tree configurations observed with RL-SPINN.

Each word i of the input sequence is represented by an embedding vector. A leaf transformation maps this vector to pair of vectors $\mathbf{r}_i^0{=}(\mathbf{h}_i^0, \mathbf{c}_i^0)$. We considered three types of leaf transformations: affine transformation, LSTM and bidirectional LSTM. The resulting representations form the initial states of the Tree-LSTM. In the absence of supervision, the tree is built in a bottom-up fashion by recursively merging consecutive constituents $(i, i+1)$ based on merge-candidate scores. On each level k of the bottom-up derivation, the merge-candidate score of the pair $(i, i{+}1)$ is computed as follow:

$$s_k(i) = \langle \mathbf{q}, \texttt{Tree-LSTM}(\mathbf{r}_i^k, \mathbf{r}_{i+1}^k) \rangle,$$

where $\mathbf{q}$ is a trainable query vector and $\mathbf{r}_i^k$ is the constituent representation at position i after k mergings. We merge a pair (i^*, i^*+1) sampled from the Categorical distribution built on the merge-candidate scores. The representations of the constituents are then updated as follow:

$$\mathbf{r}_i^{k+1} = \begin{cases} \mathbf{r}_i^k, & i < i^*, \\ \texttt{Tree-LSTM}(\mathbf{r}_i^k, \mathbf{r}_{i+1}^k) & i = i^*, \\ \mathbf{r}_{i+1}^k & i > i^*. \end{cases}$$

This procedure is repeated until one constituent remains. Its hidden state is the input sentence representation. This procedure is non-differentiable. Choi et al. (2018) use an approximation based on the Gumbel-Softmax distribution (Maddison et al., 2016; Jang et al., 2016) and the reparametrization trick (Kingma and Welling, 2013).

This relaxation makes the problem differentiable at the cost of a bias in the gradient estimates (Jang et al., 2016). This difference between the real objective function and their approximation could explain why their method cannot recover simple context-free grammars (Nangia and Bowman, 2018). We investigate this question by proposing an alternative optimisation scheme that directly aims for the correct objective function.

3 Our model

We consider the problem defined in Eq. (2) to jointly learn a composition function and an internal parser. Our model is composed of the parser of Choi et al. (2018) and the Tree-LSTM for the composition function. As suggested in past work (Mnih et al., 2016; Schulman et al., 2017), we added an entropy $\mathcal{H}$ over the tree distribution to the objective function:

$$\min_{\theta,\,\phi} \mathcal{L}(\theta, \phi) - \lambda \sum_x \mathcal{H}(t \mid x), \tag{3}$$

where $\lambda > 0$. This regulariser improves exploration by preventing early convergence to a suboptimal deterministic parsing strategy. The new objective function is differentiable with respect to θ, but not ϕ, the parameters of the parser. Learning θ follows the same procedure with BPTS as if the tree would be externally given.

In the rest of this section, we discuss the optimization of the parser and a cooperative training strategy to reduce the coadaptation issue.

3.1 Unbiased gradient estimation

We cast the training of the parser as a reinforcement learning problem. The parser is an agent whose reward function is the negative of the loss function defined in Eq. (2). Its action space is the space of binary trees. The agent's policy is a probability distribution over binary trees that decomposes as a sequence of K merging actions:

$$p_\phi(t|x) = \prod_{k=0}^{K} \pi_\phi(a_k|\mathbf{r}^k), \tag{4}$$

where $\mathbf{r}^k = (\mathbf{r}_0^k, \ldots, \mathbf{r}_{K-k}^k)$. The loss function is optimized with respect to ϕ with REINFORCE (Williams, 1992). REINFORCE requires a considerable number of random samples to obtain a gradient estimate with a reasonable level of variance. This number is positively correlated with the size of the search space, which is exponentially large in the case of binary trees. We consider several extensions of REINFORCE to circumvent this problem.

Variance reduction. An alternative solution to increasing the number of samples is the control variates method (Ross, 1997). It takes advantage of random variables with known expected values and positive correlation with the quantity whose expectation is tried to be estimated. Given an input-output pair (x, y) and tree t sampled from $p_\phi(t|x)$, let's define the random variable G as:

$$G(t) = \ell(f_\theta(x, t), y) \frac{\partial \log p_\phi(t|x)}{\partial \phi}. \quad (5)$$

According to REINFORCE, calculating the gradient with respect to ϕ for the pair (x, y) is then equivalent to determining the unknown mean of the random variable $G(t)$[2]. Let's assume there is a control variate, i.e., a random variable $b(t)$ that positively correlates with G and has known expected value with respect to $p_\phi(.|x)$. Given N samples of the $G(t)$ and the control variate $b(t)$, the new gradient estimator is:

$$G_{\text{CV}} = \mathbb{E}_{p_\phi(t|x)}[b(t)] + \frac{1}{N}\left[\sum_{i=1}^{N}(G(t_i) - b(t_i))\right].$$

A popular control variate, or baseline, used in REINFORCE is the moving average of recent rewards multiplied by the score function (Ross, 1997):

$$b(t) = c\nabla_\phi \log p_\phi(t|x).$$

It has a zero mean under the $p_\phi(.|x)$ distribution and it positively correlates with $G(t)$.

Surrogate loss. REINFORCE often is implemented via a surrogate loss defined as follow:

$$\hat{\mathbb{E}}_t \left[r_\phi(t)\ell(f_\theta(x, t), y) \right], \quad (6)$$

where $\mathbb{E}_t$ is the empirical average over a finite batch of samples and $r_\phi(t) = \frac{p_\phi(t|x)}{p_{\phi_{\text{old}}}(t|x)}$ is the probability ratio with ϕ_{old} standing for the parameters before the update.

[2]Note that while we are computing the gradients using ℓ, we could also directly optimise the parser with respect to downstream accuracy.

Input-dependent baseline. The moving average baseline cannot detect changes in rewards caused by structural differences in the inputs. In our case, a long arithmetic expression is much harder to parse than a short one, systematically leading to their lower rewards. This structural differences in the rewards aggravate the credit assignment problem by encouraging REINFORCE to discard actions sampled for longer sequences even though there might be some subsequences of actions that produce correct parsing subtrees.

A solution is to make the baseline input-dependent. In particular, we use the self-critical training (SCT) baseline of Rennie et al. (2017), defined as:

$$b(t, x) = c_{\theta,\phi}(x)\nabla_\phi \log p_\phi(t \mid x),$$

where $c_{\theta,\phi}$ is the reward obtained with the policy used at test time, i.e., $\hat{t} = \arg\max p_\phi(t|x)$. This control variate has a zero mean under the $p_\phi(t|x)$ distribution and correlates positively with the gradients. Computing the $\arg\max$ of a policy among all possible binary trees has exponential complexity. We replace it with a simpler greedy decoding, i.e, a tree t is selected by following a sequence of greedy actions $\hat{a}_k$:

$$\hat{a}_k = \arg\max \pi_\phi(a_k \mid \hat{\mathbf{r}}^k).$$

This approximation is very efficient and computing the baseline requires only one additional forward pass.

Gradient normalization. We empirically observe significant fluctuations in the gradient norms. This creates instability that can not be reduced by additive terms, such as the input-dependent baselines. A solution is to divide the gradients by a coarse approximation of their norm, e.g., a running estimate of the reward standard deviation (Mnih and Gregor, 2014). This trick ensures that the rewards remain approximately in the unit ball, making the learning process less sensitive to steep changes in the loss.

3.2 Synchronizing syntax and semantics learning with PPO

The gradients of the loss function from the Eq. (3) are calculated using two different schemes, BPST for the composition function parameters θ and REINFORCE for the parser parameters ϕ. Then, both are updated with SGD. The estimate of the gradient with respect to ϕ has higher variance compared

to the estimate with respect to θ. Hence, using the same learning rate schedule does not necessarily correspond to the same real pace of learning. It is ϕ parameters that are harder to optimise, so to improve training stability and convergence it is reasonable to aim for such updates that does not change the policy too much or too little. A simple yet effective solution is the Proximal Policy Optimization (PPO) of Schulman et al. (2017). It considers the next surrogate loss:

$$\hat{\mathbb{E}}_t \left[\max \left\{ r_\phi(t)\ell\left(f_\theta(x,t),y\right), r^c_\phi(t)\ell\left(f_\theta(x,t),y\right) \right\} \right],$$

Where $r^c_\phi(t) = \texttt{clip}\left(r_\phi(t), 1-\epsilon, 1+\epsilon\right)$ and ϵ is a real number in $(0; 0.5]$. The first argument of the $\max$ is the surrogate loss for REINFORCE. The clipped ratio in the second argument disincentivises the optimiser from performing updates resulting in large tree probability changes. With this, the policy parameters can be optimised with repeated K steps of SGD to ensure a similar "pace" of learning between the parser and the compositional function.

4 Related work

Beside the works mentioned in Sec. 2 and Sec. 3, there is a vast literature on learning latent parsers. Early connectionist work in inferring context-free grammars proposed stack-augmented models and relied on explicit supervision on the strings that belonged to the target language and those that did not (Giles et al., 1989; Sun, 1990; Das et al., 1992; Mozer and Das, 1992). More recently, new stack-augmented models were shown to learn latent grammars from positive evidence alone (Joulin and Mikolov, 2015). Our work departs from these approaches in that we aim at learning a latent grammar in the context of performing some given task. Most work that, like ours, aims at discovering the constituency structure of the input sentence relies on merging pairs of consecutive constituents. Socher et al. (2011) uses a surrogate auto-encoder objective and merges nodes greedily based on the reconstruction loss. Maillard et al. (2017) defines a relaxation of a CYK-like chart parser that is trained for a particular task. Similar idea is introduced in Le and Zuidema (2015) where an automatic parser prunes the chart to reduce the overall complexity of the algorithm. A strategy, similar in nature, has been recently proposed by Corro and Titov (2018), where Gumbel noise is used with differentiable dynamic programming to generate dependency trees. In contrast, Yogatama et al. (2016) learns a Shift-Reduce parser using reinforcement learning. Maillard and Clark (2018) further proposes a beam search strategy to overcome learning trivial trees. On a different vein, Vlad Niculae (2018) proposes a quadratic penalty term over the posterior distribution of non-projective dependency trees to enforce sparsity of the relaxation. Finally, there is a large body of work in Reinforcement Learning that aims at discovering how to combine elementary modules to solve complex tasks (Singh, 1992; Chang et al., 2018; Sahni et al., 2017). Due to the limited space we will not discuss them in further details.

Model	Accuracy
LSTM*	71.5±1.5
RL-SPINN*	60.7±2.6
Gumbel Tree-LSTM*	57.6±2.9
Ours	**99.2±0.5**

Table 1: Accuracy on the ListOps dataset. All models have 128 dimensions. Results for models with * are taken from Nangia and Bowman (2018).

5 Experiments

We conducted experiments on three different tasks: evaluating mathematical expressions on the ListOps dataset (Nangia and Bowman, 2018), sentiment analysis on the SST dataset (Socher et al., 2013a) and natural language inference task on the SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018b) datasets.

Technical details. For ListOps, we follow the experimental protocol of Nangia and Bowman (2018), i.e., a 128 dimensional model and a ten-way softmax classifier. However, we replace their multi-layer perceptron (MLP) by a linear classifier. The validation set is composed of 1k examples randomly selected from the training set. For SST and NLI, we follow the setup of Choi et al. (2018): we initialize the word vectors with GloVe300D (Pennington et al., 2014) and train an MLP classifier on the sentence representations. The hyperparameters are selected on the validation set using 5 random seeds for each configuration. Our hyperparameters are the learning rate, weight decay, the regularization parameter λ, the leaf transformations, variance reduction hyperpa-

rameters and the number of updates K in PPO. We use an adadelta optimizer (Zeiler, 2012).

5.1 ListOps

The ListOps dataset probes the syntax learning ability of latent tree models (Nangia and Bowman, 2018). It is designed to have a single correct parsing strategy that a model must learn in order to succeed. It is composed of prefix arithmetic expressions and the goal is to predict the numerical output associated with the evaluation of the expression. The sequences are made of integers in $[0, 9]$ and 4 operations: `MIN`, `MAX`, `MED` and `SUM_MOD`. The output is an integer in the range $[0, 9]$. For example, the expression `[MIN 2 [MAX 0 1] [MIN 6 3 ] 5 ]` is mapped to the output `1`. The ListOps task is thus a sequence classification problem with 10 classes. There are 90k training examples and 10k test examples. It is worth mentioning that the underlying semantic of operations and symbols is not provided. In other words, a model has to infer from examples that `[MIN 0 1] = 0`.

As shown in Table 1, the current leading latent tree models are unable to learn the correct parsing strategy on ListOps (Nangia and Bowman, 2018). They even achieve performance worse than purely sequential recurrent networks. On the other hand, our model achieves near perfect accuracy on this task, suggesting that our model is able to discover the correct parsing strategy. Our model differs in several ways from the Gumbel Tree-LSTM of Choi et al. (2018) that could explain this gap in performance. In the rest of this section, we perform an ablation study on our model to understand the importance of each of these differences.

Impact of the baseline and PPO. We report the impact of our design choices on the performance in Table 2. Our model without baseline nor PPO is vanilla REINFORCE. The baselines only improve performance when PPO is used. Furthermore, these ablated models without PPO perform on-par with the RL-SPINN model (see Table 1). This confirms our expectations for models that fail to synchronise syntax and semantics learning.

Interestingly, using PPO has a positive impact for both baselines, but accuracy remains low with the moving average baseline. The reduction of variance induced by the SCT baseline leads to a near-perfect recovery of the good parsing strategy in all five experiments. This shows the importance of this baseline for the stability of our approach.

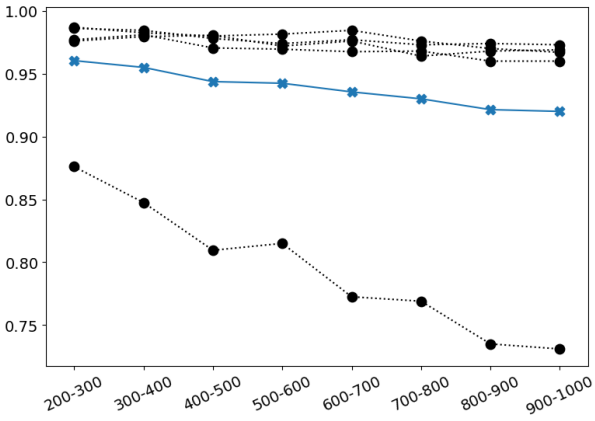

Figure 1: Blue crosses depict an average accuracy of five models on the test examples that have lengths within certain range. Black circles illustrate individual models.

Sensitivity to hyperparameters. Our model is relatively robust to hyperparameters changes when we use the SCT baseline and PPO. For example, changing the leaf transformation or dimensionality of the model has a minor impact on performance. However, we have observed that the choice of the optimiser has a significant impact. For example, the average performance drops to 73.0% if we replace Adadelta by Adam (Kingma and Ba, 2014). Yet, the maximum value out of 5 runs remains relatively high, 99.0%.

Untied parameters. As opposed to previous work, the parameters of the parser and the composition function are not tied in our model. Without this separation between syntax and semantics, it would be impossible to update one module without changing the other. The gradient direction is then dominated by the low variance signal from the semantic component, making it hard to learn the parser. We confirm experimentally that our model with tied parameters fails to find the correct parser and its accuracy drops to 64.7%.

Extrapolation and Grammaticality. Recursive models have the potential to generalise to any sequence length. Our model was trained with sequences of length up to 130 tokens. We test the ability of the model to generalise to longer sequences by generating additional expressions of lengths 200 to 1000. As shown in Fig.1, our model has a little loss in accuracy as the length increases to ten times the maximum length seen

	No baseline		Moving average		Self critical	
	No PPO	PPO	No PPO	PPO	No PPO	PPO
min	61.7	61.4	61.7	59.4	63.7	**98.2**
max	70.1	76.6	74.3	96.0	64.1	**99.6**
mean $\pm$ std	66.2 $\pm$3.2	66.5 $\pm$5.9	65.5 $\pm$ 4.7	67.5 $\pm$14.3	64.0 $\pm$0.1	**99.2 $\pm$0.5**

Table 2: Accuracy on ListOps test set for our model with three different baselines, with and without PPO. We use $K = 15$ for PPO.

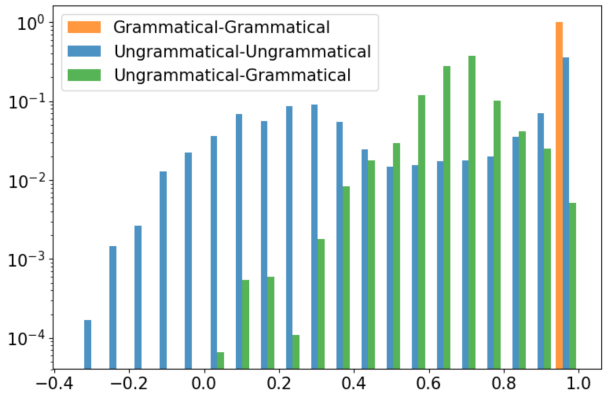

Figure 2: The distributions of cosine similarity for elements from the different sets of mathematical expressions. A logarithmic scale is used for y-axis.

Model	Dim.	Acc.
Yogatama et al. (2016)	100	80.5
Maillard et al. (2017)	100	81.6
Choi et al. (2018)	100	82.6
Ours	100	**84.3$\pm$0.3**
Bowman et al. (2016)	300	83.2
Munkhdalai and Yu (2017)	300	84.6
Choi et al. (2018)	300	**85.6**
Choi et al. (2018)†	300	83.7
Choi et al. (2018)*	300	84.9$\pm$ 0.1
Ours	300	85.1$\pm$0.2
Chen et al. (2017)	600	85.5
Choi et al. (2018)	600	**86.0**
Ours	600	84.6$\pm$0.2

Table 3: Results on SNLI. *: publicly available code and hyperparameter optimization was used to obtain results. †: results are taken from Williams et al. (2018a)

during training.

On the other hand, we notice that final representations produced by the parser are very similar to each other. Indeed, the cosine similarity between these vectors for the test set has a mean value of 0.998 with a standard deviation of 0.002. There are two possible explanations for this observation: either our model assigns similar representations to valid expressions, or it produces a trivial uninformative representation regardless of the expression. To verify which explanation is correct, we generate ungrammatical expressions by removing either one operation token or one closing bracket symbol for each sequence in the test set. As shown in Figure 2, in contrast to grammatical expressions, ungrammatical ones tend to be very different from each other: "Happy families are all alike; every unhappy family is unhappy in its own way." The only exception, marked by a mode near 1, come from ungrammatical expressions that represent incomplete expressions because of missing a closing bracket at the end. This kind of sequences were seen by the parser during training and they indeed have to be represented by the same vector. These observations show that our model does not produce a trivial representation, but identifies the rules and constraints of the grammar. Moreover, vectors for grammatical sequences are so different from vectors for ungrammatical ones that you can tell them apart with 99.99% accuracy by simply measuring their cosine similarity to a randomly chosen grammatical vector from the training set. Interestingly, we have not observed a similar signal from the vectors generated by the composition function. Even learning a naive classifier between grammatical and ungrammatical expressions on top of these representations achieves an accuracy of only 75%. This suggests that most of the syntactic information is captured by the parser, not the composition function.

5.2 Natural Language Inference

We next evaluate our model on natural language inference using the Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) and

Model	Dim.	Acc.
LSTM†	300	69.1
SPINN†	300	67.5
RL-SPINN†	300	67.4
Gumbel Tree-LSTM†	300	69.5
Ours	300	**70.7±0.3**

Table 4: Results on MultiNLI. †: results are taken from Williams et al. (2018a).

MultiNLI (Williams et al., 2018b) datasets. Natural language inference consists in predicting the relationship between two sentences which can be either entailment, contradiction, or neutral. The task can be formulated as a three-way classification problem. The results are shown in Tables 3 and 4. When training the model on MultiNLI dataset we augment the training data with the SNLI data and use *matched* versions of the development and test sets. Surprisingly, two out of four models for MultiNLI task collapsed to left-branching parsing strategies.This collapse can be explained by the absence of the entropy regularization and the small number of PPO updates $K = 1$, which were determined to be optimal via hyperparameter optimization. As with ListOps, using an Adadelta optimizer significantly improves the training of the model.

5.3 Sentiment Analysis

We evaluate our model on a sentiment classification task using the Stanford Sentiment Treebank (SST) of Socher et al. (2013a). All sentences in SST are represented as binary parse trees, and each subtree of a parse tree is annotated with the corresponding sentiment score. There are two versions of the dataset, with either binary labels, "negative" or "positive", (SST-2) or five labels, representing fine-grained sentiments (SST-5). As shown in Table 5, our results are in line with previous work, confirming the benefits of using latent syntactic parse trees instead of the predefined syntax.

We noticed that all models trained on NLI or sentiment analysis tasks have parsing policies with relatively high entropy. This indicates that the algorithm does not prefer any specific grammar. Indeed, generated trees are very similar to balanced ones. This result is in line with Shi et al. (2018) where they observe that binary balanced tree encoder gets the best results on most classification tasks.

	SST-2	SST-5
Sequential sentence representation		
Radford et al. (2017)	**91.8**	52.9
McCann et al. (2017)	90.3	53.7
Peters et al. (2018)	-	**54.7**
RvNN based models with external tree		
Socher et al. (2013a)	85.4	45.7
Tai et al. (2015)	88.0	51.0
Munkhdalai and Yu (2017)	89.3	53.1
Looks et al. (2017)	89.4	52.3
RvNN based models with latent tree		
Yogatama et al. (2016)	86.5	-
Choi et al. (2018)	90.7	53.7
Choi et al. (2018)*	90.3±0.5	51.6±0.8
Ours	90.2±0.2	51.5±0.4

Table 5: Accuracy results of models on the SST. All the numbers are from Choi et al. (2018) but * where we used their publicly available code and performed hyperparameter optimization.

We also compare with state-of-the-art sequence-based models. For the most part, these models are pre-trained on larger datasets and fine-tuned on these tasks. Nonetheless, they outperform recursive models by a significant margin. Performance on these datasets is more impacted by pre-training than by learning the syntax. It would be interesting to see if a similar pre-training would also improve the performance of recursive models with latent tree learning.

6 Conclusion

In this paper, we have introduced a novel model for learning latent tree parsers. Our approach relies on a separation between syntax and semantics. This allows dedidcated optimization schemes to each modules. In particular, we found that it is important to have a unbiased estimator of the parser gradients and to allow multiple gradient steps with PPO. Our learned parser generalizes to sequences of any length and distinguishes grammatical from ungrammatical expressions by forming meaningful representations for well-formed expressions. Additionally, our approach performs competitively on several real natural language tasks. In the future, we would like to further explore relaxation-based techniques for

learning the parser, such as REBAR (Tucker et al., 2017) or ReLAX (Grathwohl et al., 2017). Finally, we plan to look into applying recursive approaches to language modeling as a pre-training step and measure if it has the same impact on downstream tasks as sequential models.

Acknowledgments

We would like to thank Alexander Koller, Ivan Titov, Wilker Aziz and anonymous reviewers for their helpful suggestions and comments.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings EMNLP 2015*, pages 632–642.

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 ACL 2016, Volume 1: Long Papers*.

Michael B. Chang, Abhishek Gupta, Sergey Levine, and Thomas L. Griffiths. 2018. Automatically composing representation transformations as a means for generalization. *CoRR*, abs/1807.04640.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of ACL 2017, Volume 1: Long Papers*, pages 1657–1668.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of AAAI 2018*.

Caio Corro and Ivan Titov. 2018. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *CoRR*, abs/1807.09875.

Sreerupa Das, C Lee Giles, and Guo-Zheng Sun. 1992. Learning context-free grammars: Capabilities and limitations of a recurrent neural network with an external stack memory. In *Proceedings of CogSci 1992*, page 14.

Andrew Drozdov and Samuel Bowman. 2017. The coadaptation problem when learning how and what to compose. *Proceedings of the 2nd Workshop on Representation Learning for NLP*.

C. Lee Giles, Guo-Zheng Sun, Hsing-Hen Chen, Yee-Chun Lee, and Dong Chen. 1989. Higher order recurrent networks and grammatical inference. In *Proceedings of NIPS 1989*, pages 380–387.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*, 1:347–352.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David K. Duvenaud. 2017. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *CoRR*, abs/1711.00123.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proceedings of NIPS 2015*, pages 190–198.

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

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Phong Le and Willem H. Zuidema. 2015. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of EMNLP 2015*, pages 1155–1164.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. *arXiv preprint arXiv:1702.02181*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712.

Jean Maillard and Stephen Clark. 2018. Latent tree learning with differentiable parsers: Shift-reduce parsing and chart parsing. *CoRR*, abs/1806.00840.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *CoRR*, abs/1705.09189.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS 2017*, pages 6294–6305.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of ICML 2016*, pages 1928–1937.

Michael Mozer and Sreerupa Das. 1992. A connectionist symbol manipulator that discovers the structure of context-free languages. In *Proceedings of NIPS 1992*, pages 863–870.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *Proceedings of EACL 2017*, volume 1, page 11. NIH Public Access.

Nikita Nangia and Samuel R. Bowman. 2018. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of NAACL-HLT 2018, Student Research Workshop*, pages 92–99.

Barbara BH Partee, Alice G ter Meulen, and Robert Wall. 1990. *Mathematical methods in linguistics*, volume 30. Springer Science & Business Media.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of CVPR 2017*, pages 1179–1195.

Sheldon M. Ross. 1997. *Simulation (2. ed.)*. Statistical modeling and decision science. Academic Press.

Himanshu Sahni, Saurabh Kumar, Farhan Tejani, and Charles L. Isbell. 2017. Learning to compose skills. *CoRR*, abs/1711.11289.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. *CoRR*, abs/1808.09644.

Satinder P. Singh. 1992. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of ICML 2011*, pages 129–136.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013a. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*, pages 1631–1642.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*, pages 1631–1642.

G Sun. 1990. Connectionist pushdownautomata that learn context-free gram-mars. In *Proc. IJCNN'90*, volume 1, pages 577–580.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL 2015, Volume 1: Long Papers*, pages 1556–1566.

George Tucker, Andriy Mnih, Chris J. Maddison, John Lawson, and Jascha Sohl-Dickstein. 2017. REBAR: low-variance, unbiased gradient estimates for discrete latent variable models. In *Proceedings of NIPS 2017*, pages 2624–2633.

Claire Cardie Vlad Niculae, André F. T. Martins. 2018. Towards dynamic computation graphs via sparse latent structure. *CoRR*, abs/1809.00653.

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

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018b. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

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

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of ICML 2015*, pages 1604–1612.