Latent Tree Learning with Differentiable Parsers: Shift-Reduce Parsing and Chart Parsing

Jean Maillard, Stephen Clark
Computer Laboratory, University of Cambridge
`jean@maillard.it, sc609@cam.ac.uk`

Abstract

Latent tree learning models represent sentences by composing their words according to an induced parse tree, all based on a downstream task. These models often outperform baselines which use (externally provided) syntax trees to drive the composition order. This work contributes (a) a new latent tree learning model based on shift-reduce parsing, with competitive downstream performance and non-trivial induced trees, and (b) an analysis of the trees learned by our shift-reduce model and by a chart-based model.

1 Introduction

Popular recurrent neural networks in NLP, such as the Gated Recurrent Unit (Cho et al., 2014) and Long Short-Term Memory (Hochreiter and Schmidhuber, 1997), compute sentence representations by reading their words in a sequence. In contrast, the Tree-LSTM architecture (Tai et al., 2015) processes words according to an input parse tree, and manages to achieve improved performance on a number of linguistic tasks.

Recently, Yogatama et al. (2016), Maillard et al. (2017), and Choi et al. (2017) all proposed sentence embedding models which work similarly to a Tree-LSTM, but do not require any parse trees as input. These models function without the assistance of an external automatic parser, and without ever being given any syntactic information as supervision. Rather, they induce parse trees by training on a downstream task such as natural language inference. At the heart of these models is a mechanism to assign trees to sentences – effectively, a natural language parser. Williams et al. (2017a) have recently investigated the tree structures induced by two of these models, trained for

a natural language inference task. Their analysis showed that Yogatama et al. (2016) learns mostly trivial left-branching trees, and has inconsistent performance; while Choi et al. (2017) outperforms all baselines (including those using trees from conventional parsers), but learns trees that do not correspond to those of conventional treebanks.

In this paper, we propose a new latent tree learning model. Similarly to Yogatama et al. (2016), we base our approach on shift-reduce parsing. Unlike their work, our model is trained via standard backpropagation, which is made possible by exploiting beam search to obtain an approximate gradient. We show that this model performs well compared to baselines, and induces trees that are not as trivial as those learned by the Yogatama et al. model in the experiments of Williams et al. (2017a).

This paper also presents an analysis of the trees learned by our model, in the style of Williams et al. (2017a). We further analyse the trees learned by the model of Maillard et al. (2017), which had not yet been done, and perform evaluations on both the SNLI data (Bowman et al., 2015) and the MultiNLI data (Williams et al., 2017b). The former corpus had not been used for the evaluation of trees of Williams et al. (2017a), and we find that it leads to more consistent induced trees.

2 Related work

The first neural model which learns to both parse a sentence and embed it for a downstream task is by Socher et al. (2011). The authors train the model's parsing component on an auxiliary task, based on recursive autoencoders, while the rest of the model is trained for sentiment analysis.

Bowman et al. (2016) propose the "Shift-reduce Parser-Interpreter Neural Network", a model which obtains syntax trees using an integrated shift-reduce parser (trained on gold-

standard trees), and uses the resulting structure to drive composition with Tree-LSTMs.

Yogatama et al. (2016) is the first model to jointly train its parsing and sentence embedding components. They base their model on shift-reduce parsing. Their parser is not differentiable, so they rely on reinforcement learning for training.

Maillard et al. (2017) propose an alternative approach, inspired by CKY parsing. The algorithm is made differentiable by using a soft-gating approach, which approximates discrete candidate selection by a probabilistic mixture of the constituents available in a given cell of the chart. This makes it possible to train with backpropagation.

Choi et al. (2017) use an approach similar to easy-first parsing. The parsing decisions are discrete, but the authors use the Straight-Through Gumbel-Softmax estimator (Jang et al., 2017) to obtain an approximate gradient and are thus able to train with backpropagation.

Williams et al. (2017a) investigate the trees produced by Yogatama et al. (2016) and Choi et al. (2017) when trained on two natural language inference corpora, and analyse the results. They find that the former model induces almost entirely left-branching trees, while the latter performs well but has inconsistent trees across re-runs with different parameter initializations.

A number of other neural models have also been proposed which create a tree encoding during parsing, but unlike the above architectures rely on traditional parse trees. Le and Zuidema (2015) propose a sentence embedding model based on CKY, taking as input a parse forest from an automatic parser. Dyer et al. (2016) propose RNNG, a probabilistic model of phrase-structure trees and sentences, with an integrated parser that is trained on gold standard trees.

3   Models

CKY   The model of Maillard et al. (2017) is based on chart parsing, and effectively works like a CKY parser (Cocke, 1969; Kasami, 1965; Younger, 1967) using a grammar with a single nonterminal A with rules $A \rightarrow A\ A$ and $A \rightarrow \alpha$, where α is any terminal. The parse chart is built bottom-up incrementally, like in a standard CKY parser. When ambiguity arises, due to the multiple ways to form a constituent, all options are computed using a Tree-LSTM, and scored. The constituent is then represented as a weighted sum

of all possible options, using the normalised scores as weights. In order for this weighted sum to approximate a discrete selection, a temperature hyperparameter is used in the softmax. This process is repeated for the whole chart, and the sentence representation is given by the topmost cell.

We noticed in our experiments that the weighted sum still occasionally assigned non-trivial weight to more than one option. The model was thus able to utilize multiple inferred trees, rather than a single one, which would have potentially given it an advantage over other latent tree models. Hence for fairness, in our experiments we replace the softmax-with-temperature of Maillard et al. (2017) with a softmax followed by a straight-through estimator (Bengio et al., 2013). In the forward pass, this approach is equivalent to an argmax function; while in the backward pass it is equivalent to a softmax. Effectively, this means that a single tree is selected during forward evaluation, but the training signal can still propagate to every path during backpropagation. This change did not noticeably affect performance on development data.

Beam Search Shift-Reduce   We propose a model based on beam search shift-reduce parsing (BSSR). The parser works with a queue, which holds the embeddings for the nodes representing individual words which are still to be processed; and a stack, which holds the embeddings of the nodes which have already been computed. A standard binary Tree-LSTM function (Tai et al., 2015) is used to compute the d-dimensional embeddings of nodes:

$$\begin{bmatrix} \boldsymbol{i} \\ \boldsymbol{f}_L \\ \boldsymbol{f}_R \\ \boldsymbol{u} \\ \boldsymbol{o} \end{bmatrix} = \mathbf{W}\boldsymbol{w} + \mathbf{U}\boldsymbol{h}_L + \mathbf{V}\boldsymbol{h}_R + \boldsymbol{b},$$

$$\boldsymbol{c} = \boldsymbol{c}_L \odot \sigma(\boldsymbol{f}_L) + \boldsymbol{c}_R \odot \sigma(\boldsymbol{f}_R) + \tanh(\boldsymbol{u}) \odot \sigma(\boldsymbol{i}),$$

$$\boldsymbol{h} = \sigma(\boldsymbol{o}) \odot \tanh(\boldsymbol{c}),$$

where $\mathbf{W}, \mathbf{U}$ are learned $5d \times d$ matrices, and $\boldsymbol{b}$ is a learned $5d$ vector. The d-dimensional vectors $\sigma(\boldsymbol{i}), \sigma(\boldsymbol{f}_L), \sigma(\boldsymbol{f}_R)$ are known as *input* gate and *left-* and *right-forget* gates, respectively. $\sigma(\boldsymbol{o}_t)$ and $\tanh(\boldsymbol{u}_t)$ are known as *output* gate and *candidate update*. The vector $\boldsymbol{w}$ is a word embedding, while $\boldsymbol{h}_L, \boldsymbol{h}_R$ and $\boldsymbol{c}_L, \boldsymbol{c}_R$ are the childrens' $\boldsymbol{h}$- and $\boldsymbol{c}$-states. At the beginning, the queue contains embeddings for the nodes corresponding to

single words. These are obtained by computing the Tree-LSTM with w set to the word embedding, and $h_{L/R}, c_{L/R}$ set to zero. When a SHIFT action is performed, the topmost element of the queue is popped, and pushed onto the stack. When a REDUCE action is performed, the top two elements of the stack are popped. A new node is then computed as their parent, by passing the children through the Tree-LSTM, with $w = 0$. The new node is then pushed onto the stack.

Parsing actions are scored with a simple multi-layer perceptron, which looks at the top two stack elements and the top queue element:

$$r = \mathbf{W}_{s1} \cdot h_{s1} + \mathbf{W}_{s2} \cdot h_{s2} + \mathbf{W}_q \cdot h_{q1},$$
$$p = \text{softmax} \left(a + \mathbf{A} \cdot \tanh r \right),$$

where h_{s1}, h_{s2}, h_{q1} are the h-states of the top two elements of the stack and the top element of the queue, respectively. The three $\mathbf{W}$ matrices have dimensions $d \times d$ and are learned; a is a learned 2-dimensional vector; and $\mathbf{A}$ is a learned $2 \times d$ vector. The final scores are given by $\log p$, and the best action is greedily selected at every time step. The sentence representation is given by the h-state of the top element of the stack after $2n - 1$ steps.

In order to make this model trainable with gradient descent, we use beam search to select the b best action sequences, where the score of a sequence of actions is given by the sum of the scores of the individual actions. The final sentence representation is then a weighted sum of the sentence representations from the elements of the beam. The weights are given by the respective scores of the action sequences, normalised by a softmax and passed through a straight-through estimator. This is equivalent to having an argmax on the forward pass, which discretely selects the top-scoring beam element, and a softmax in the backward pass.

4 Experimental Setup

Data To match the settings of Maillard et al. (2017), we run experiments with the SNLI corpus (Bowman et al., 2015). We additionally run a second set of experiments with the MultiNLI data (Williams et al., 2017b), and to match Williams et al. (2017a) we augment the MultiNLI training data with the SNLI training data. We call this augmented training set *MultiNLI+*. For the MultiNLI+ experiments, we use the *matched* versions of the development and test sets. We use

Model	SNLI	MultiNLI+
Prior work: Baselines		
100D LSTM (Yogatama)	80.2	—
300D LSTM (Williams)	82.6	69.1
100D Tree-LSTM (Yogatama)	78.5	—
300D SPINN (Williams)	82.2	67.5
Prior work: Latent Tree Models		
100D ST-Gumbel (Choi)	81.9	—
300D ST-Gumbel (Williams)	83.3	**69.5**
300D ST-Gumbel[†] (Williams)	**83.7**	67.5
100D CKY (Maillard)	81.6	—
100D RL-SPINN (Yogatama)	80.5	—
300D RL-SPINN[†] (Williams)	82.3	67.4
This work: Latent Tree Models		
100D CKY (Ours)	82.2	69.1
100D BSSR (Ours)	83.0	69.0

Table 1: SNLI and MultiNLI (matched) test set accuracy. †: results are for the model variant without the leaf RNN transformation.

pre-trained 100D GloVe embeddings[1] (Pennington et al., 2014) for performance reasons, and fine-tune them during training. Unlike Williams et al. (2017a), we do not use a bidirectional leaf transformation. Models are optimised with Adam (Kingma and Ba, 2014), and we train five instances of every model. For BSSR, we use a beam size of 50, and let it linearly decrease to its final size of 5 over the first two epochs.

Setup To assign the labels of *entails*, *contradicts*, or *neutral* to the pairs of sentences, we follow Yogatama et al. (2016) and concatenate the two sentence embeddings, their element-wise product, and their squared Euclidean distance into a vector v. We then calculate $q = \text{ReLU} \left(\mathbf{C} \cdot v + c \right)$, where $\mathbf{C}$ is a $200 \times 4d$ learned matrix and c a 200-dimensional learned bias; and finally predict $p(y = c \mid q) \propto \exp \left(\mathbf{B} \cdot q + b \right)$ where $\mathbf{B}$ is a 3×200 matrix and b is 3-dimensional.

5 Experiments

For each model and dataset, we train five instances using different random initialisations, for a total of $2 \times 2 \times 5 = 20$ instances.

NLI Accuracy We measure SNLI and MultiNLI test set accuracy for CKY and BSSR. The aim is to ensure that they perform reasonably, and are in line with other latent tree learning models of a similar size and complexity. Results for the best mod-

---

[1] https://nlp.stanford.edu/projects/glove/

Dataset	Model	Self-F1	Left Branching μ (σ)	max	Right Branching μ (σ)	max	Stanford Parser μ (σ)	max
					F1 w.r.t.			
MultiNLI+	300D SPINN (Williams)	71.5	19.3 (0.4)	19.8	36.9 (3.4)	**42.6**	**70.2** (3.6)	**74.5**
MultiNLI+	300D ST-Gumbel (Williams)	49.9	32.6 (2.0)	35.6	**37.5** (2.4)	40.3	23.7 (0.9)	25.2
MultiNLI+	300D ST-Gumbel† (Williams)	41.2	30.8 (1.2)	32.3	35.6 (3.3)	39.9	27.5 (1.0)	29.0
MultiNLI+	300D RL-SPINN† (Williams)	**98.5**	**99.1** (0.6)	**99.8**	10.7 (0.2)	11.1	18.1 (0.1)	18.2
MultiNLI+	100D CKY (Ours)	45.9	32.9 (1.9)	35.1	31.5 (2.3)	35.1	23.7 (1.1)	25.0
MultiNLI+	100D BSSR (Ours)	46.6	40.6 (6.5)	47.6	24.2 (6.0)	27.7	23.5 (1.8)	26.2
MultiNLI+	*Random Trees* (Williams)	32.6	27.9 (0.1)	27.9	28.0 (0.1)	28.1	27.0 (0.1)	27.1
SNLI	100D RL-SPINN (Yogatama)	—	—	41.4	—	19.9	—	**41.7**
SNLI	100D CKY (Ours)	59.2	43.9 (2.2)	46.9	**33.7** (2.6)	**36.7**	30.3 (1.1)	32.1
SNLI	100D BSSR (Ours)	**60.0**	**48.8** (5.2)	**53.9**	26.5 (6.9)	34.0	**32.8** (3.5)	36.4
SNLI	*Random Trees* (Ours)	35.9	32.3 (0.1)	32.4	32.5 (0.1)	32.6	32.3 (0.1)	32.5

Table 2: Unlabelled F1 scores of the trees induced by various models against: other runs of the same model, fully left- and right-branching trees, and Stanford Parser trees provided with the datasets. The baseline results on MultiNLI are from Williams et al. (2017a). $\dagger$: results are for the model variant without the leaf RNN transformation.

els, chosen based on development set performance, are reported in Table 1.

While our models do not reach the state of the art, they perform at least as well as other latent tree models using 100D embeddings, and are competitive with some 300D models. They also outperform the 100D Tree-LSTM of Yogatama et al. (2016), which is given syntax trees, and match or outperform 300D SPINN, which is explicitly trained to parse.

Self-consistency Next, we examine the consistency of the trees produced for the development sets. Adapting the code of Williams et al. (2017a), we measure the models' *self F1*, defined as the unlabelled F1 between trees by two instances of the same model (given by different random initializations), averaged over all possible pairs. Results are shown in Table 2. In order to test whether BSSR and CKY learn similar grammars, we calculate the *inter-model F1*, defined as the unlabelled F1 between instances of BSSR and CKY trained on the same data, averaged over all possible pairs. We find an average F1 of 42.6 for MultiNLI+ and 55.0 for SNLI, both above the random baseline.

Our Self F1 results are all above the baseline of random trees. For MultiNLI+, they are in line with ST-Gumbel. Remarkably, the models trained on SNLI are noticeably more self-consistent. This shows that the specifics of the training data play an important role, even when the downstream task is the same. A possible explanation is that MultiNLI has longer sentences, as well as multiple genres, including telephone conversations which

often do not constitute full sentences (Williams et al., 2017b). This would require the models to learn how to parse a wide variety of styles of data. It is also interesting to note that the inter-model F1 scores are not much lower than the self F1 scores. This shows that, given the same training data, the grammars learned by the two different models are not much more different than the grammars learned by two instances of the same model.

F1 Scores Finally, we investigate whether these models learn grammars that are recognisably left-branching, right-branching, or similar to the trees produced by the Stanford Parser which are included in both datasets. We report the unlabelled F1 between these and the trees from from our models in Table 2, averaged over the five model instances. We show mean, standard deviation, and maximum.

We find a slight preference from BSSR and the SNLI-trained CYK towards left-branching structures. Our models do not learn anything that resembles the trees from the Stanford Parser, and have an F1 score with them which is at or below the random baseline. Our results match those of Williams et al. (2017a), which show that whatever these models learn, it does not resemble PTB grammar.

6 Conclusions

First, we proposed a new latent tree learning model based on a shift-reduce parser. Unlike a previous model based on the same parsing technique, we showed that our approach does not learn triv-

ial trees, and performs competitively on the downstream task.

Second, we analysed the trees induced by our shift-reduce model and a latent tree model based on chart parsing. Our results confirmed those of previous work on different models, showing that the learned grammars do not resemble PTB-style trees (Williams et al., 2017a). Remarkably, we saw that the two different models tend to learn grammars which are not much more different than those learned by two instances of the same model.

Finally, our experiments highlight the importance of the choice of training data used for latent tree learning models, even when the downstream task is the same. Our results suggest that MultiNLI, which has on average longer sentences coming from different genres, might be hindering the current models' ability to learn consistent grammars. For future work investigating this phenomenon, it may be interesting to train models using only the written genres parts of MultiNLI, or MultiNLI without the SNLI corpus.

Acknowledgments

References

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

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

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)*, pages 1466–1477, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Learning to compose task-specific tree structures. *arXiv*, abs/1707.02786.

John Cocke. 1969. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University.

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*, pages 199–209, San Diego, California. Association for Computational Linguistics.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax.

T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

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

Phong Le and Willem Zuidema. 2015. 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, Lisbon, Portugal. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory 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)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2017a. Learning to parse from a semantic objective: It works. is it syntax?

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017b. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv*, abs/1704.05426.

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

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.