LaTeX Exercises for DS-GA 1012

Adina Williams[1]
adinawilliams@nyu.edu

Andrew Drozdov[2,*]
andrew.drozdov@nyu.edu

Samuel R. Bowman[1,2,3]
bowman@nyu.edu

[1]Dept. of Linguistics
New York University
10 Washington Place
New York, NY 10003

[2]Dept. of Computer Science
New York University
60 Fifth Avenue
New York, NY 10011

[3]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

Abstract

This document contains various bits and pieces pulled from a recent paper from NYU. Don't try to read it—plenty of important parts were removed to speed up compilation time. If you're curious, the actual paper is here: https://arxiv.org/abs/1709.01121. **To complete the exercises:** Open the TeX source for this document and look for the string "`% Exercise`".

1 Introduction

Tree-structured recursive neural networks (TreeRNNs; Socher et al., 2011)—which build a vector representation for a sentence by incrementally computing representations for each node in its parse tree—have been proven to be effective at sentence understanding tasks like sentiment analysis (Socher et al., 2013), textual entailment (Bowman et al., 2016), and translation (Eriguchi et al., 2016). Some variants of these models (Socher et al., 2011; Bowman et al., 2016) can also be trained to produce parse trees that they then consume. Recent work on *latent tree learning* (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2017) has led to the development of new training methods for TreeRNNs that allow them to learn to parse without ever being given an example of a correct parse tree, replacing direct syntactic supervision with indirect supervision from a downstream task like sentence classification. These models are designed to learn grammars—strategies for assigning trees to sentences—that are suited to help solve the sentence understanding task at hand, rather than ones that approximate expert-designed grammars like that of the Penn Treebank (PTB; Marcus et al., 1999).

[*]Now at eBay, Inc.

2 Models and Methods

This paper investigates the behavior of two models: RL-SPINN and ST-Gumbel. Both have been shown to outperform similar models based on supervised parsing, and the two represent two substantially different approaches to latent tree learning.

Additional Details We implement all models in PyTorch 0.2. We closely follow the original Theano code for SPINN in our implementation, and we incorporate source code provided by Choi et al. for the core parsing data structures and sampling mechanism of the ST-Gumbel model. Our code, saved models, and model output are available on GitHub.[1]

We use GloVe vectors to represent words (standard 300D 840B word package, without fine tuning; Pennington et al., 2014), and feed them into a 2×300D bi-directional GRU RNN (based on the *leaf LSTM* of Choi et al.) to give the models access to local context information when making parsing decisions. To understand the impact of this component, we follow Choi et al. in also training each model with the leaf GRU replaced with a simpler context-sensitive input encoder that simply multiplies each GloVe vector by a matrix. We find that these models perform best when the temperature of the ST-Gumbel distribution is a trained parameter, rather than fixed at 1.0 as in Choi et al..

We use L2 regularization and apply dropout (Srivastava et al., 2014) to the input of the 1024D sentence pair combination layer. We train all models using the Adam optimizer (Kingma and Ba, 2015). For hyperparameters for which no obvious default value exists—the L2 and dropout parameters, the relative weighting of the gradients from REINFORCE in RL-SPINN, the starting

[1]https://github.com/nyu-mll/spinn/tree/is-it-syntax-release

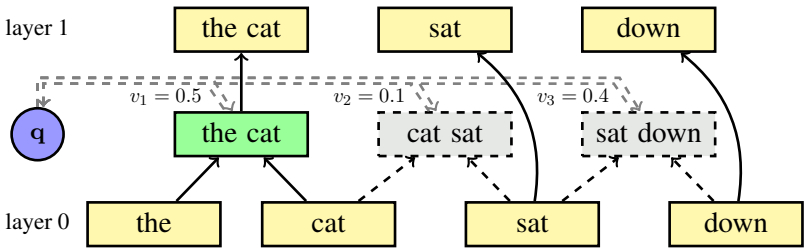

Figure 1: The ST-Gumbel model in its first step of processing the sentence *the cat sat down*, based on a figure by Choi et al. (2017), used with permission. The model first computes a composed representation for every pair of adjacent words or phrases (shown with dotted lines), assigns each of these a score v_i, and then uses these to sample a discrete variable q which determines which representation is preserved for the next layer.

learning rate, and the size of the tracking LSTM state in SPINN—we heuristically select ranges in which usable values can be found (focusing on MultiNLI development set performance), and then randomly sample values from those ranges. We train each model five times using different samples from those ranges and different random initializations for model parameters. We use early stopping based on development set performance with all models.

3 Analyzing the Learned Trees

In the previous section, we have shown that latent tree learning models are able to perform as well or better than models that have access to linguistically principled parse trees at training or test time, but that the grammars that they learn are neither consistent across runs, nor meaningfully similar to PTB grammar. In this section, we investigate the trees produced by these learned grammars directly to identify whether they capture any recognizable syntactic or semantic phenomena.

The RL-SPINN models create overwhelmingly left-branching trees. We observe few deviations from this pattern, but these occur almost exclusively on sentences of fewer than seven words.

In some preliminary tuning runs not shown above, we saw models that deviated from this pattern more often, and one that fixated on *right*-branching structures instead, but we find no grammatically interesting patterns in any of these deviant structures.

The ST-Gumbel models learned substantially more complex grammars, and we focus on these for the remainder of the section. We discuss three model behaviors which yield linguistically implausible constituents. The first two highlight settings where the ST-Gumbel model is consistent where it shouldn't be, and the third highlights a setting in which it is worryingly inconsistent. The models' treatment of these three phenomena and our observation of these models' behavior more broadly suggest that the models do not produce

Model	**SNLI**	**MNLI**
Prior Work: Baselines		
100D LSTM (Yogatama)	80.2	–
100D TreeLSTM (Yogatama)	78.5	–
300D SPINN (Bowman)	83.2	–
300D SPINN-PI-NT (Bowman)	80.9	–
300D BiLSTM (Williams)	81.5	67.5
Prior Work: Latent Tree Learning		
100D RL-SPINN (Yogatama)	80.5	–
100D Soft Gating (Maillard)	81.6	–
100D ST-Gumbel (Choi)	81.9	–
w/o Leaf LSTM	80.2	–
300D ST-Gumbel (Choi)	84.6	–
w/o Leaf LSTM	82.2	–
600D ST-Gumbel (Choi)	**85.4**	–
This Work: Baselines		
300D SPINN	81.9	66.9
w/o Leaf GRU	82.2	67.5
300D SPINN-NC	81.6	68.1
w/o Leaf GRU	82.4	67.8
300D SPINN-PI-NT	81.9	68.2
w/o Leaf GRU	81.7	67.6
This Work: Latent Tree Learning		
300D ST-Gumbel	83.3	**69.5**
w/o Leaf GRU	83.7	67.5
300D RL-SPINN	81.7	67.3
w/o Leaf GRU	82.3	67.4

Table 1: Test set results. Our implementations of SPINN and RL-SPINN differ only in parser design, and our implementations of SPINN-NC and ST-Gumbel differ only in parser design. SPINN-PI-NT includes no tracking or parsing component and uses Stanford Parser trees at test time.

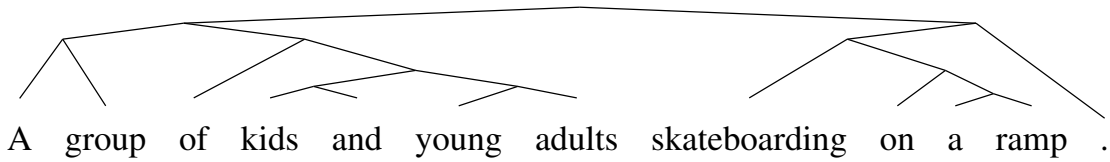

Figure 2: Ground truth tree for a premise sentence in SNLI's training dataset.

trees that follow any recognizable semantic or syntactic principles.

4 Conclusion

The experiments and analysis presented in this paper show that the best available models for latent tree learning learn grammars that do not correspond to the structures of formal syntax and semantics in any recognizable way. In spite of this, these models perform as well or better on sentence understanding—as measured by MultiNLI performance—as models with access to Penn Treebank-style parses.

This result leaves us with an immediate puzzle: What do these models—especially those based on the ST-Gumbel technique—learn that allows them to do so well? We have presented some observations, but we are left without a fully satisfying explanation. A thorough investigation of this problem will likely require a search of new architectures for sentence encoding that borrow various behaviors from the models trained in this work.

This result also opens farther-reaching questions about grammar and sentence understanding: Will the optimal grammars for sentence understanding problems like NLI—were we to explore the full space of grammars to find them—share any recognizable similarities with the structures seen in formal work on syntax and semantics? *A priori*, we should expect that they should. While it is unlikely that PTB grammar is strictly optimal for any task, the empirical motivations for many of its core constituent types—the noun phrase, the prepositional phrase, and so forth—are straightforward and compelling. However, our best latent tree learning models are not able to discover these structures.

If we accept that some form of principled constituent structure is necessary or desirable, then we are left with an engineering problem: How do we identify this structure? Making progress in this direction will likely involve both improvements to the TreeRNN models at the heart of latent tree learning systems, to make sure that these models are able to perform composition effectively enough to be able to make full use of learned structures, and also improvements to the structure search methods that are used to explore possible grammars.

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices.

References

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 (ACL)*. Association for Computational Linguistics, Berlin, Germany.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. ArXiv preprint 1707.02786.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 823–833. http://www.aclweb.org/anthology/P16-1078.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2017. Jointly learning sentence embeddings and

syntax with unsupervised tree-lstms. ArXiv preprint 1705.09189.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. LDC99T42. Linguistic Data Consortium.

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)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

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 (EMNLP)*. Association for Computational Linguistics, Edinburgh, UK, pages 151–161. http://www.aclweb.org/anthology/D11-1014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1631–1642. http://www.aclweb.org/anthology/D13-1170.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)* 15(1):1929–1958.

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