LaTeX Exercises for DS-GA 1011

Adina Williams[1]
adinawilliams@nyu.edu

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

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

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

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

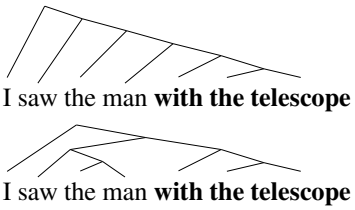
[3]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

Abstract

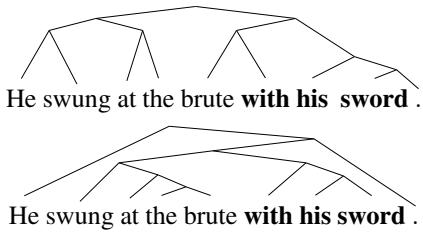
This document contains various bits and pieces pulled from a recent paper from NYU. Don't try to read it—plenty of important parts were removed to speed up compilation time. If you're curious, the actual paper is here: https://arxiv.org/abs/1709.01121. **To complete the exercises:** Open the TeX source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



(a) Two different trees lead to two different interpretations for the sentence in example (1).



(b) Parses generated by at ST-Gumbel model (left) and the Stanford Parser (right).

Figure 1: Examples of unlabeled binary parse trees.

Latent tree learning models have shown striking success at sentence understanding, reliably performing better on sentiment analysis and textual entailment than do comparable TreeRNN models which use parses assigned by conventional parsers, and setting the state of the art among sentence-encoding models for textual entailment. However, none of the work in latent tree learning to date has included any substantial evaluation of the quality of the trees induced by these models, leaving open an important question which this paper aims to answer: Do these models owe their success to consistent, principled latent grammars? If they do, these grammars may be worthy objects of study for researchers in syntax and semantics. If they do not, understanding why the models succeed without syntax could lead to new insights into the use of TreeRNNs and into sentence understanding more broadly.

[*]Now at eBay, Inc.

While there is still lively debate within linguistic syntax and semantics over the precise grammars that should be used for language understanding and generation, it has been clear since at least Chomsky (1965); Frege (1892); Heim and Kratzer (1998) that understanding any natural language sentence requires implicitly or explicitly recognizing which substrings of the sentence form meaningful units or *constituents*.

This is well illustrated by syntactically ambiguous sentences like the one below, repeated from Sag (1991) a.o.:

1. (a) I [saw the man] [with the telescope]
   ↪ I used the telescope to view the man
   (b) I saw the [man [with the telescope]]
   ↪I saw the man who had a telescope

Under the constituency parse shown in 1a, *with a telescope* provides additional information about the action described by the constituent *saw a man*, and in 1b, *with a telescope* provides additional information about the individual described by *man*. If the same string of words can be assigned two different valid constituency structures, two different interpretations generally result. Constituency information can be straightforwardly expressed using an unlabeled parse tree like the ones used in TreeRNNs, and expressing constituency information is the generally the primary motivation for using trees in TreeRNNs.

In this paper, we reimplement the latent tree learning models of Yogatama et al. (2017) and Choi et al. (2017) in a shared codebase, train both models (and several baselines) to perform textual entailment on the SNLI and MultiNLI corpora (Bowman et al., 2015; Williams et al., 2017), and evaluate the results quantitatively and qualitatively with a focus on four issues: the degree to which latent tree learning improves task performance, the degree to which latent tree learning models learn similar grammars across random restarts, the degree to which their grammars match PTB grammar, and the degree to which their grammars appear to follow any recognizable grammatical principles.

We confirm that both types of model succeed at producing useful sentence representations, but find that only the stronger of the two models—that of Choi et al. (2017)—learns a nontrivial grammar or outperforms its baseline. We find that that grammar agrees with PTB grammar with roughly chance accuracy, and does not show any other consistent, linguistically plausible patterns. We do find, though, that the resulting grammar has some regularities, including a preference for shallow trees, and a preference to treat pairs of adjacent words at the edges of a sentence as constituents.

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.

SPINN Variants All three of our baselines and one of the two latent tree learning models are based on the SPINN architecture of Bowman et al. (2016).

In the base SPINN model, all model components are used, and the transition classifier is trained on binarized Penn Treebank-style parses from the Stanford PCFG Parser (Klein and Manning, 2003), which are included with SNLI and MultiNLI. These binary-branching parse trees are converted to SHIFT/REDUCE sequences for use in the model through a simple reversible transformation.

RL-SPINN, based on the unsupervised syntax model of Yogatama et al. (2017), is architecturally equivalent to SPINN, but its transition classifier is optimized for MultiNLI classification accuracy, rather than any parsing-related loss. Because this component produces discrete decisions, the REINFORCE algorithm is used (with the standard moving average reward baseline) to supply gradients for it.

We also evaluate two other variants of SPINN. In SPINN-NC (for No Connection from tracking to composition), the connection from the tracking LSTM to the composition function is severed. This weakens the model, but makes it exactly equivalent to a plain TreeLSTM—it will produce the exact same vector that a TreeLSTM with the same composition function would have produced for the tree that the transition classifier implicitly produces. This model serves as a maximally comparable baseline for the ST-Gumbel model, which also performs composition using a standard TreeLSTM in forward-propagation.

SPINN-PI-NT (for Parsed Input, No Tracking) removes the tracking LSTM as well as the

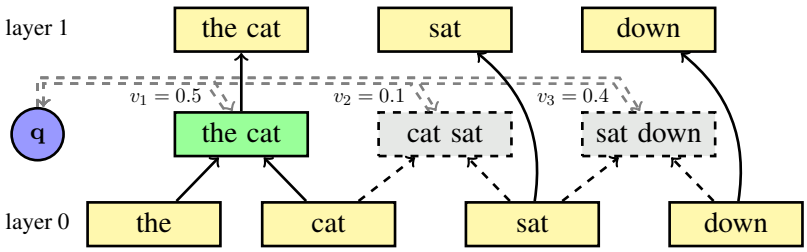


Figure 2: The ST-Gumbel model in its first step of processing the sentence *the cat sat down*, based on a figure by Choi et al. (2017), used with permission. The model first computes a composed representation for every pair of adjacent words or phrases (shown with dotted lines), assigns each of these a score v_i, and then uses these to sample a discrete variable q which determines which representation is preserved for the next layer.

two components that depend on it: the tracking-composition connection and the transition decision function. As such, it cannot produce its own parse trees, and must rely on trees from the input data. We include this in our comparison to understand the degree to which training a parser, rather than using a higher-quality off-the-shelf parser, impacts performance on our semantic task.

ST-Gumbel The ST-Gumbel model was developed by Choi et al. (2017) and is shown in Figure 2. The model takes a sequence of $N - 1$ steps to build a tree over N words. At every step, every possible pair of adjacent words or phrase vectors in the partial tree is given to a TreeLSTM composition function to produce a new candidate phrase vector. A simple learned scoring function then selects the best of these candidates, which forms a constituent node in the tree and replaces its two children in the list of nodes that are available to compose. This repeats until only two nodes remain, at which point they are composed and the tree is complete. This exhaustive search increases the computational complexity of the model over (RL-)SPINN, but also allows the model to perform a form of easy-first parsing, making it easier for the model to explore the space of possible parsing strategies.

Though the scoring function yields discrete decisions, the Straight-Through Gumbel-Softmax estimator of Jang et al. (2016) makes it possible to nonetheless efficiently compute an approximate gradient for the full model without the need for relatively brittle reinforcement learning techniques.

Data Our primary experiments use the Multi-Genre Natural Language Inference Corpus (MultiNLI; Williams et al., 2017). MultiNLI is a 433k-example textual entailment dataset created in the style of SNLI, but with a more diverse range of source texts and longer and more complex sentences, which we expect will encourage the models to produce more consistent and interpretable trees than they otherwise might. Following Williams et al., we train on the combination of MultiNLI and SNLI in these experiments (yielding just under 1M training examples) and evaluate on MultiNLI (using the *matched* development and test sets).

We also evaluate trained models on the full Wall Street Journal section of the Penn Treebank, a seminal corpus of manually-constructed constituency parses which introduced the parsing standard used in this work. Because the models under study produce and consume binary-branching constituency trees without labels (and because such trees are already included with SNLI and MultiNLI), we use the Stanford Parser's `CollapseUnaryTransformer` and `TreeBinarizer` tools to convert these Penn Treebank Trees to this form.

Sentence Pair Classification Because our textual entailment task requires a model to classify *pairs* of sentences, but the models under study produce vectors for single sentences, we concatenate the two sentence vectors, their difference, and their elementwise product (following Mou et al., 2016), and feed the result into a 1024D ReLU layer to produce a representation for the sentence pair. This representation is fed into a three-way softmax classifier that selects one of the labels *entailment*, *neutral*, and *contradiction* for the pair.

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 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 Does latent tree learning help sentence understanding?

Table 1 shows the accuracy of all models on two test sets: SNLI (training on SNLI only, for comparison with prior work), and MultiNLI (training on both datasets). Each figure represents the accuracy of the best run, selected using the development set, of five runs with different random initializations and hyperparameter values.

On both SNLI and MultiNLI, we reproduce the key result of Choi et al., showing that the ST-Gumbel model, which receives no syntactic information at training time, outperforms SPINN-NC, which performs composition in an identical manner but is explicitly trained to parse. This suggests that the latent trees are helpful for constructing semantic representations for sentences, whether or not they resemble conventional parse trees.

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.

Our results with RL-SPINN are more equivocal. That model matches, but does not beat, the performance of the full SPINN model, which is equivalent except that it is trained to parse. However, our implementation of RL-SPINN outperforms Yogatama et al.'s (lower-dimensional) implementation by a substantial margin. The impact of the leaf GRU is sometimes significant, but the direction of this effect is not consistent.

Our results with SPINN-PI-NT are not substantially better than those with any other model, suggesting the relatively simple greedy parsing strategies used by the other models are not a major limiting factor in their performance. None of our

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

Model	F1 wrt. Left Branching μ (σ)	max	F1 wrt. Right Branching μ (σ)	max	F1 wrt. Stanford Parser μ (σ)	max	Microavg. Depth μ (σ)	max
300D SPINN	19.3 (0.4)	19.8	36.9 (3.4)	**42.6**	70.2 (3.6)	**74.5**	6.2 (0.2)	6.5
w/o Leaf GRU	21.2 (1.4)	22.9	39.0 (2.6)	41.4	63.5 (1.7)	65.7	6.4 (0.1)	6.5
300D SPINN-NC	19.2 (0.4)	19.7	36.2 (1.4)	38.2	70.5 (2.0)	73.1	6.1 (0.0)	6.1
w/o Leaf GRU	20.6 (0.5)	21.3	38.9 (2.5)	41.9	64.1 (2.7)	67.1	6.3 (0.0)	6.3
300D ST-Gumbel	32.6 (2.0)	35.6	37.5 (2.4)	40.3	23.7 (0.9)	25.2	4.1 (0.1)	4.2
w/o Leaf GRU	30.8 (1.2)	32.3	35.6 (3.3)	39.9	27.5 (1.0)	29.0	4.6 (0.1)	4.7
300D RL-SPINN	95.0 (1.4)	96.6	13.5 (1.8)	15.8	18.8 (0.2)	19.0	8.6 (0.0)	**8.6**
w/o Leaf GRU	99.1 (0.6)	**99.8**	10.7 (0.2)	11.1	18.1 (0.1)	18.2	8.6 (0.0)	**8.6**
Random Trees	27.9 (0.1)	27.9	28.0 (0.1)	28.1	27.0 (0.1)	27.1	4.4 (0.0)	4.4

Table 2: *F1 wrt.* shows F1 scores on the MultiNLI development set with respect to strictly right- and left-branching trees and with respect to the Stanford Parser trees supplied with the corpus. *Microavg. Depth* shows the average across sentences of the average depth of each word in its tree. Each is shown with the mean, standard deviation, and maximum of the metric across the five runs of each model.

models reach the state of the art on either task, but all are comparable in both absolute and relative performance to other published results, suggesting that we have trained reasonable examples of latent tree learning models and can draw informative conclusions by studying the behaviors of these models.

4 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 trees that follow any recognizable semantic or syntactic principles.

5 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, 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*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642. http://aclweb.org/anthology/D15-1075.

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.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT press.

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.

Gottlob Frege. 1892. Über sinn und bedeutung. *Wittgenstein Studien* 1(1).

Irene Heim and Angelika Kratzer. 1998. *Semantics in generative grammar*. Blackwell.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Sapporo, Japan, pages 423–430. http://www.aclweb.org/anthology/P03-1054.

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.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 130–136. https://doi.org/10.18653/v1/P16-2022.

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.

Ivan A. Sag. 1991. Linguistic theory and natural language processing. In *Natural language and speech: symposium proceedings, Brussels*. Springer, pages 69–83.

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.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. ArXiv preprint 1704.05426. http://arxiv.org/abs/1704.05426.

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)*.