Capstone Mid-Semester Memo: Learning to Parse without Being Taught How

Nikita Nangia Student

Samuel Bowman Advisor

Michael Gill Instructor

1 Original Proposal

What is the project?
"Investigating whether the bias in ST-Gumbel softmax forces suboptimal decisions in Learning to Compose Task-Specific Tree Structures by Choi et al. 2017. Furthermore, testing out the possibility of always investigating all possible parse trees so as to increase the probability of choosing for optimal tree structures."

What is the proposed scope of the project?
"To first gain a strong understanding of the possible pitfalls of the ST-Gumbel softmax method for latent tree learning. Then write a new less-efficient but potentially more effective deep learning model (in PyTorch) that explores all parse trees during training, and analyzing the ways in which this model differs in its behavior from ST-Gumbel."

2 Research Question

Tree Structure Tree-structured recursive neural networks (TreeRNN; Socher et al., 2011) have effectively been used to build sentence representation for tasks like sentiment analysis (Socher et al., 2013) and textual entailment (Bowman et al., 2016). The thinking behind TreeRNNs being that languages inherently have structure, and the parse of a sentence offers information about that structure. Providing the parse of a sentence to a model, and having it use this structural information while building sentence representation, should yield more informative and accurate semantic representations.

Latent Learning In this research project we're looking specifically at learning to parse without giving the true parse. In this setting, without any syntactic supervision, the system will rely on the semantic downstream task to guide it in building informative parsing structure. There has been recent work on this topic, and we're most closely looking at Choi et al.'s (2017) and Maillard et al.'s (2017) methods. In the Choi et al. method, the system always makes hard parsing decisions using ST-Gumbel to choose the composition at each layer. While this allows the model to train very quickly, in about 2 hours on the Stanford Natural Language Inference corpus (SNLI; Bowman et al., 2015), the method also leads an unusual bias wherein the model backpropogates to invalid trees, thus making it harder to learn informative parsing. The Maillard et al. approach avoids such a bias by using chart parsing, but they show worse results on the SNLI test set, and state that their model takes 5 days to converge.

Our hope is to firstly build a system that uses chart parsing as in Maillard et al. (2017), but with a significantly shaved down training time. Secondly, to study the kinds of parses the system settles on and to see if they are more informative than those chosen by the Choi et al. model. Since a model using chart parsing explores all possible valid trees, by taking a linear combination of all possible compositions at each layer, we hope that such a model will learn more interesting and informative parsing structures than the Choi et al. model. As was shown in Williams et al. (2017a), the Choi et al. settles on very shallow and trees and seem to learn grammars that are quite different from Penn Treebank (PTB; Marcus et al., 1999) grammars.

3 Data

The task we've chosen to study our method on is textual entailment. It was shown in Conneau

et al. (2017), that training a model to do textual entailment, aka natural language inference, yields relatively more universal sentence representations. It is therefore a good task to use to judge the more general applicability of our approach. The two corpora we are using are SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017b). SNLI is the standard corpus used to study textual entailment, and MultiNLI, while similar, is a more diverse and challenging corpus.

4 Methodology

The models we've discussed so far, and the idea we've presented, are all deep learning models. We're building our system in PyTorch. The Choi et al., Maillard et al., and our model will all be in a single codebase[1] to get comparable results. The Williams et al. (2017a) paper released PyTorch code for the Choi et al. model.[2] Our code is forked from there and the remaining systems will be built on top of it.

We first need to write and replicate the Maillard et al. model and their results. Then we can make alterations to the model, proceed with studying the parse trees, seeing what we can infer, and decide where to go next.

5 Discussion

5.1 Progress and current status

Reproduce So far, the Maillard et al. model has been written and the accuracy we see on the SNLI test set is within 0.5% of the original paper. There were some hiccups along the way; getting chart parsing to work correctly took a couple weeks.

Accuracy and time We made the switch from standard standard softmax to Gumbel softmax to weight the compositions at each layer. The final accuracy of the model is currently 81.2% on the SNLI test set (Maillard et al. report 81.6%), and the model converges in about a day instead of 5! So we've at least managed to shave down training time significantly.

Trees Now we can focus on the interesting part, which is the actual parsing structure. We ran a few experiments. First, the standard model which uses Gumbel softmax to weigh compositions. We also tried uniform weighting, random weighting, and weighting using ST-Gumbel softmax. The accuracy results aren't significantly different and are shown in Table 1. More importantly, the parsing structure doesn't look notably different at first glance. The trees for all the chart-parsing models are quite shallow like the Choi et al. model. These results are preliminary and entirely qualitative since we haven't yet run full statistics on the parses yet. It is disconcerting though that the random weighting might be just as effectual (or ineffectual) as the Gumbel or ST-Gumbel approach.

Model Type	SNLI
ST-Gumbel (Choi)	85.4
Chart-parsing with Gumbel softmax	81.2
Chart-parsing with ST-Gumbel	81.6
Chart-parsing with Uniform	80.5
Chart-parsing with Random	81.0

Table 1: Test set results of various models. All models use 600 dimensions, and 300D GloVe vectors (Pennington et al., 2014). The four chart-parsing models are identical except for the function that weighs the compositions and gives scores for their linear combination.

Discussion One important thing to note is that all experiments thus far use a "leaf encoder", dubbed by Choi et al.. This simply means that a standard bidirectional RNN (with GRU units) is used to first encode the sentence thus giving the TreeRNN the information about where the sentence begins and ends. This step may be inflating the results and hiding any existing issues with chart parsing. Additionally, even if chart-parsing is perfectly set up, we know that bidirectional RNN are already very good sentence encoders on their own, so using the leaf encoder could be burying signal that would help the model learn how to parse.

Another point to note is that we are not doing temperature annealing yet. It was previously found that with the Choi et al. model, temperature annealing was not helpful, but this is technique to test out in future experiments since it was used in Maillard et al. (2017). Currently, to get parse trees for the Gumbel softmax and random weighting approach, we're choosing the series of compositions with the highest weight.

[1] github.com/NYU-CDS-Capstone-Project/Betelgeuse_SPINN

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

5.2 Next steps

Immediate The most pressing next steps are to get full statistics on the parse trees we have, to run experiments without the leaf encoder, and try temperature annealing. If the results continue to be consistent with what we've seen so far, i.e. if the parse trees are indeed very shallow and their isn't much difference between random and Gumbel/ST-Gumbel weighting, and accuracy doesn't suddenly fall several percentage points without the leaf encoder, then we will probe the parsing decisions made by the model.

Analyzing parse trees We will want to study the parse trees first on an artificial dataset where a model needs to be able to intelligently parse to succeed, meaning a dataset on which a standard RNN would fail. We will construct a dataset which will be composed on numbers and a series of arithmetic operations, e.g. the sequence $(+\ (\max\ (5\ 9)\ 11))$ will have the answer 20. There will be some trial and error and creating a dataset that RNNs fail on but that we can do with chart-parsing since chart-parsing is computationally intensive and we can't use very long sequences. Once we have this dataset, we can test to see if TreeRNN method with chart-parsing succeeds, and if it does, what kinds of parse trees does it find. On studying these trees, we can begin to understand what kind of bias is in the model and whether we can account for it.

More baslines One additional and important experiment that needs to be run is to force the model to make the correct parsing decision and see how it performs. This is an ablation study which will both test the code to make sure it is working as expected, but also be an important baseline to compare against.

In conclusion, there is still a ways to go for this project! But we're now at the interesting bit and will hopefully soon have more insight on whether this style of model can learn to parse without direct supervision.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proc. 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 *Proc. ACL*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *CoRR* http://arxiv.org/abs/1707.02786.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the EMNLP*. http://arxiv.org/abs/1705.02364.

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

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*.

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*.

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 EMNLP*. http://www.aclweb.org/anthology/D13-1170.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2017a. Learning to parse from a semantic objective: It works. is it syntax? *CoRR* http://arxiv.org/abs/1709.01121.

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