Chart Parsing for Latent Tree Learning

Nikita Nangia[1]
nikitanangia@nyu.edu

Samuel R. Bowman[1]
bowman@nyu.edu

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

Abstract

Latent tree learning models have shown promising results at sentence understanding. They consistently outperform fully supervised TreeRNN models on sentence classification tasks like sentiment analysis and textual entailment. In spite of these results however, analysis of the parses produced by these models has shown that they do not learn a consistent, informative strategy, but simply build shallow, random trees (Williams et al., 2017). These existing models learn grammars that do not correspond to the structures of formal syntax and semantics in any recognizable way. In this work, we explore a model architecture that is designed to address some of the pitfalls of previous latent tree learning models. We present a model that uses a CYK-style chart parser (Cocke, 1969; Younger, 1967; Kasami, 1965) to learn to parse without direct supervision, while using the produced parses to build sentence representations. We show that this model, unlike previous latent tree learning models, is capable of learning informative grammars and consistent parsing strategies.

1 Introduction

It is has been clear for a while that natural language is structured and understanding natural language sentences requires recognizing which substrings of the sentence compose to form meaningful subunits Chomsky (1965); Frege (1892); Heim and Kratzer (1998). Sentences can be syntactically ambiguous and different parses of the same sentences can result in different meanings. This is illustrated with an example in Figure 1. Given the first constituency parse, the phrase *with the telescope* informs the constituent *saw the man*, thus implying that, "Using a telescope, I saw the man". While with the second constituency parse, *with the telescope* describes the man being seen, implying that, "I saw a man who had a telescope." Since the same sentence, under different parses, can have different meaning, we can conclude that using the structure of sentences should be useful in building semantically accurate representations of those sentences.

Tree-structured recursive neural networks (TreeRNNs; Socher et al., 2011) do exactly this by using a sentence's parse to build a vector representation of a sentence. They do this by computing representations for each node in the parse, and incrementally composing them to get the final sentence representation. It has been shown that this style of neural network is effective at sentence understanding tasks like textual entailment (Bowman et al., 2016), sentiment analysis (Socher et al., 2013), and translation (Eriguchi et al., 2016). TreeRNNs perform better than standard RNNs on these tasks, indicating that using the parsing structure of sentence does enable the model to build a better representation, and to better capture the true meaning and semantics of language.

There has been some work showing that such TreeRNN based models can also be trained to produce parses that they then consume to build sentence representations (Socher et al., 2011; Bowman et al.,

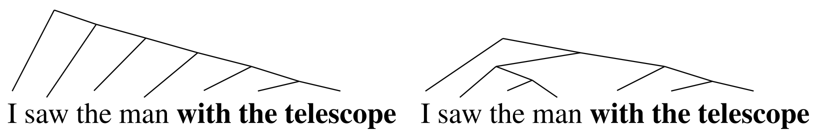
I saw the man **with the telescope**
I saw the man **with the telescope**

Figure 1: Two different trees lead to two different interpretations for the same sentence. This example is borrowed from Williams et al. (2017).

2016). In more recent work on *latent tree learning* (Yogatama et al., 2017; Maillard et al., 2017; Choi et al., 2017) there have been efforts made to enable these models to learn to parse without direct supervision for parsing, but only indirect supervision from the downstream task. Such models do not require access to the true, gold parse of any given sentence; they instead rely on the signal from the downstream task, like sentence classification, to train the model's parser. Since these models aren't guided by true parses, they are theoretically able to explore and learn grammars best suited to solve the sentence understanding task at hand. As a result, we wouldn't expect their learnt grammars to identically reflect expert-designed grammars like that of the Penn Treebank (PTB; Marcus et al., 1999). However, Williams et al. (2017) have shown that the current best latent tree learning models, notably including Choi et al. (2017), do not learn interesting, interpretable grammars. Though these models outperform TreeRNNs that use gold parses from conventional parsers, the latent tree learning models simply learn to build shallow random trees, reducing the distance between the final sentence representation and the word representation, thus assisting in gradient flow.

In this paper, we will be presenting an architecture that is closely modeled on one presented by Maillard et al. (2017), using the CYK parsing algorithm (Cocke, 1969; Younger, 1967; Kasami, 1965). We implement changes to their model to enable improvements in the parsing choices of the model. We ultimately show that, unlike with existing latent tree learning models, our model is capable of learning to sensibly parse and learn informative grammars.

2 Related Work

This work is builds on existing work in latent tree learning. In particular, we build on the work done by Choi et al. (2017) and Maillard et al. (2017). Both these papers present latent tree learning models that perform better than TreeRNNs on sentence classification tasks. Maillard et al.'s 2017 model use the CYK parsing algorithm to make parsing deicisions. In their model, each node is a weighted sum of all candidate compositions, and therefore the final sentence representation is a linear combination of all valid parse trees. This model is computational quite expensive, an the authors report a training time of one week on the Stanford Natural Language Inference corpus (SNLI; Bowman et al., 2015).

Choi et al.'s 2017 model however employs a strategy in which their model makes discrete, hard choices while building compositions; this model trains very quickly, in abour 2 hours on SNLI. While the Choi et al. model is the best current latent tree learning model, its final graph results in an unusual quirk in which the model backpropogates to the chosen parse tree and to a multitude of invalid parse trees, for example trees in which the same word is composed more than once. As a result, the parses chosen by this model are simply random, shallow trees.

3 Model Description

The model we present leverages the advantages of both Maillard et al. (2017) and Choi et al. (2017). It uses a CYK style parser to learn to parse without direct supervision, it makes either soft or hard choices during composition, and it uses a tree-structured long short-term memory RNN (Tree-LSTM; Tai et al., 2015; Zhu et al., 2015) to build compositions. The version of our model that makes hard parsing decision cuts down on training time.

3.1 Tree-LSTM

Tree-LSTMs are an extension to of the LSTM (Hochreiter and Schmidhuber, 1997) architecture to tree structures. A Tree-LSTM controls the flow of information from children to parent nodes. It uses the cell state to compute a parent representation from its children, thus allowing the cell to capture

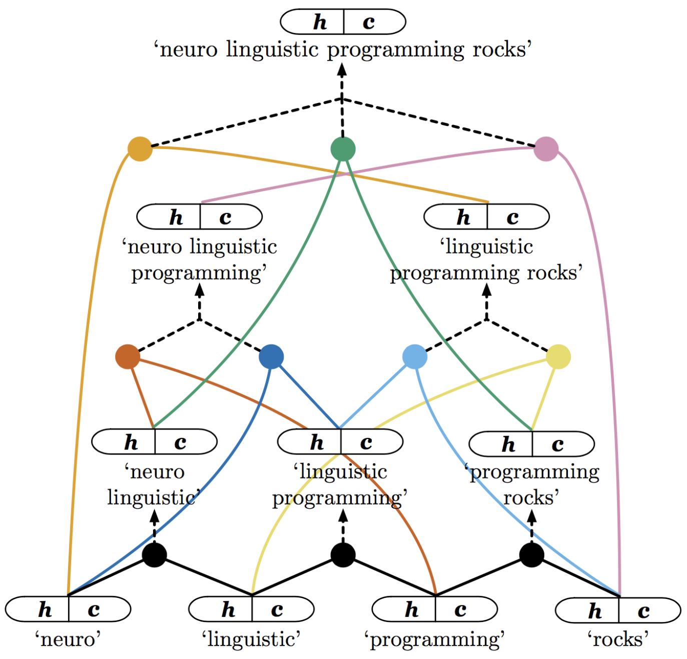

Figure 2: Model architecture when making soft decision at the combination stage. This figure is borrowed from Maillard et al. (2017).

distant vertical dependencies. The Tree-LSTM has the distinct advantage of having a relatively short distance between the word level leaf nodes, to the final sentence representation at the root.

Given a node with left and right children labelled L and R respectively, the sentence representation is computed as follows,

$$\begin{bmatrix} \mathbf{i} \\ \mathbf{f_L} \\ \mathbf{f_R} \\ \mathbf{o} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}_{comp} \begin{bmatrix} \mathbf{h_l} \\ \mathbf{h_r} \end{bmatrix} + \mathbf{b}_{comp} \right) \tag{1}$$

$$\mathbf{c_p} = \mathbf{f_L} \odot \mathbf{c_L} + \mathbf{f_R} \odot \mathbf{c_R} + \mathbf{i} \odot \mathbf{u} \tag{2}$$

$$\mathbf{h_p} = \mathbf{o} \odot \tanh(\mathbf{c_p}) \tag{3}$$

where $\mathbf{W}_{comp} \in \mathbb{R}^{5d \times 2d}$, and $\mathbf{b}_{comp} \in \mathbb{R}^{2d}$ (d is dimensional the size of the hidden layer), and $\odot$ is the element-wise product. The above computations are repeated recursively following the tree structure provided by the sentence's parse, and the representation of the whole sentence is given by the $\mathbf{h}$-state of the root node.

3.2 CYK Parser

The CYK parser represents all possible binary parse trees for a given sentence, according to some grammar. As a result, we can use this chart data structure to store and build all possible binary-branching trees. The only pre-requisite to building a chart parse is being given the word, leaf-level nodes. The subsequent compositions can be computed by following the chart structure.

Figure 2 shows an example of how we use the CYK parsing strategy. Given any two constituents, individual tokens or composition of tokens, a composition with neighboring constituents if computed and stored. When there is ambiguity in how to compute a constituent, a linear combination of possible compositions is taken. For example, to get a representation of "neuro linguistic programming" in

Figure 2, the model can either compose “neuro” with “linguistic programming”, or compose “neuro linguistic” with “programming”. To resolve the ambiguity, both possible compositions are scored using a scoring vector by taking a dot product,

$$e_i = \mathbf{h} \cdot \mathbf{v} \tag{4}$$

where $\mathbf{h}_i$ is the representation of the composition, v is the scoring vector which is a learned parameter if dimension d, and e_i is the assigned scalar score. Once all possible compositions are scored, we normalize them with Gumbel-Softmax (Jang et al., 2016) and then take a weighted sum. In Maillard et al. (2017), they normalize using a simple softmax before calculating the weighted sum.

3.3 Gumbel-Softmax

Soft Trees Gumbel-Softmax approximates one-hot vectors by making them continuous, it can therefore be used with discrete random variables in a network. When we have a selection of possible compositions for a single constituent, we are confronted with discrete random variables. We use Gumbel-Softmax since it can be trained with standard backpropogation and has shown to have lower variance than score-based gradient estimators like REINFORCE (Williams, 1992).

To compute a Gumbel-Softmax, we sample first sample Gumbel noise $g_1, \dots, g_k \sim$ Gumbel(0,1)[2]. Then given the unnormalized probabilites, $e_1, \dots, e_k$, Gumbel-Softmax is given by

$$s_i = \frac{\exp((\log(e_i) + g_i)/\tau)}{\sum_j \exp((\log(e_j) + g_j)/\tau)} \tag{5}$$

where τ is a tunable temperature parameter. As τ goes to zero, the the Gumbel-Softmax estimator becomes more peaky and starts to closely resemble a one-hot vector.

Once we have the normalized scores from the Gumbel-Softmax, to compute soft trees, we take a weighted sum of all the scored compositions to get the parent cell and hidden states thus,

$$\mathbf{c}_p = \sum_{i=1}^{k} s_i \mathbf{c}_i, \qquad \mathbf{h}_p = \sum_{i=1}^{k} s_i \mathbf{h}_i \tag{6}$$

This linear combination version of our model, building soft trees, is most similar to Maillard et al. (2017). The key difference being that we use Gumbel-Softmax since it is demonstrably better at approximating one-hot vectors.

Hard trees In contrast, to cut down on training time, and to leverage the advantages of the Choi et al. (2017) model, we also build a model which uses the Straight-Through (ST) Gumbel-Softmax estimator (Bengio et al., 2013) to make discrete choices. In this estimator, the standard Gumbel-Softmax is computed, and in the forward pass it discretizes the continuous sample $(s_1, \dots, d_k)$ to a one hot vector by finding the $\arg\max$,

$$s_i^{ST} = \begin{cases} 1 & i = \arg\max_j s_j \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

In the backward pass, it still uses the already computed continuous Gumbel-Softmax. The ST Gumbel-Softmax estimator was used in Choi et al. (2017) to make all composition decision, while we use a mixture of CYK parsing the ST Gumbel-Softmax.

Model	**SNLI Acc.**	**Avg. Depth**
Prior Work: Baselines		
300D BiLSTM	81.5	-
300D SPINN	83.2	-
100D Chart-parsing (Maillard)	81.6	-
300D ST-Gumbel (Choi)	**84.6**	4.2
This Work: Chart-Parsing		
300D Random Trees	82.4	**4.71**
300D Gumbel (soft)	82.5	4.02
300D ST-Gumbel (hard)	82.9	4.44

Table 1: Accuracy results on SNLI dev set and the average tree depth for latent tree learning models.

4 Experiments

We evaluate the proposed model on natural language data and on a pre-fix arithmetic dataset. The code for our implementation in Pytorch is publicly available [1].

4.1 Natural Language

For natural language, we choose the task of textual entailment, or Natural Language Inference (NLI). We choose this task because NLI is an effective task to measure a model's ability at natural language understanding (Conneau et al., 2017). Furthermore, the existing latent tree leanring models that we are comparing our model against have been evaluated on SNLI (Bowman et al., 2015).

In NLI, a model is presented with two sentences, a premise and hypothesis, and it must determine the relationship between the two sentences. In SNLI, the possible relations are entailment, contradiction, or neither. So we built a Siamese architecture with separate parameters for the premise and hypothesis sentences. Given the representations for both sentences, $\mathbf{h}^p$ and $\mathbf{h}^h$, we concatenate the two sentences, their element-wise product, and their difference. This is then passed to a two-layer multi-layer perceptron (MLP) to get the predicted relation.

Training We choose the standard size of 300-D for the hidden states. We use 300-D pre-trained GloVe vectors (Pennington et al., 2014), and we dp not tune them during training. A mini-batch size of 128 is used during training, and the MLP has a 1048-D hidden layer. When training the "soft" and "random" versions of our model, we do not do a scheduled annealing of the temperature parameter.

Results We train three version of our model. The first does soft combinations of compositions by only using Gumbel-Softmax. The second makes hard decisions in the forward pass by using ST-Gumbel Softmax. Lastly, we build a model which selects random trees by randomly assigning weights to the compositions, and then calculating the weighted sum. Table 1 shows accuracy results for all three models and relevant baseline models. For baseline comparison, we use the SPINN model (Bowman et al., 2016), which does directly supervised learning of the parser, the Choi et al. (2017), and Maillard et al. (2017) models. We find that while our model does outperform Maillard et al.'s, it still lags behind the latent tree learning state of the art model by Choi et al..

Notably though, while our "soft" model is just as computationally intensive as Maillard et al.'s, our model trains considerably faster. Instead of their one week training time, our model trains on SNLI in under two days.

Parsing quality Even though our model has lower accuracy on sentence classification with SNLI, it does show interesting results in the quality of parses it choses. It was shown in Williams et al. (2017) that the Choi et al. model ultimately does not learn any proper grammar and learns to select random, shallow trees. To analyze the quality of parses our model produces, we gathered a few

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

		Model Type	
	Random	**ST-Gumbel**	**Gumbel**
Accuracy on Node Types			
NP	30.5	**43.5**	37.3
PP	14.4	**15.6**	14.7
SBAR	**5.6**	4.5	2.4
ADJP	18.1	21.0	**24.8**
ADVP	15.6	**19.5**	18.8
FRAG	94.9	94.9	**96.6**
F1 Score with Tree Types			
Left branching	32.4	39.3	**43.4**
Right branching	32.2	30.0	**32.8**
Ground truth	32.1	**38.0**	32.8

Table 2: The top of the table shows accuracy on correctly composing various node types on SNLI test set. The bottom shows F1 scores against left branching, right branching, and ground truth trees on SNLI test set.

statistics shown in Table 2. As we can see, our model finds a relatively sensible parsing strategy. Unlike the Choi et al. model, our model has a significantly higher F1-score with ground truth trees than the random trees do, indicating that the chosen parses are not strictly random. The ST-Gumbel version of our model performs best on this metric. Furthermore, as shows in Table 1, our model builds deeper trees than the Choi et al. model; however the trees are not significantly deeper and this is not a compelling result.

We do see that our models have highest F1-scores with left-branching trees. This is a curious result since the English language, and in SNLI, sentences tend to be right-branching. This indicates that while the model is building more informative trees than simply random trees, it still isn't finding grammars that are consistent with our understanding of English.

In Table 2 we also see that our models significantly outperform the "Random" model in correctly composing non-terminal nodes. Williams et al. (2017) showed that Choi et al.'s 2017 model performs at par with random trees on these metric. Therefore, we have compelling evidence to suggest that our model is indeed learning to parse intelligently.

In Figure 3 we show two examples of how our ST-Gumbel chart parsing model parses sentences, compared to traditional parses from the Stanford-parser. We can see that while our model does not produce the same parse, it does learn to compose sensible constituents in the sentences. The word *jean* and *short* are correctly paired, as are *soccer* and *ball*. The phrase *The little boy in jean shorts* is also composed into a single constituent, which is finally composed with *kicks the soccer ball .* to create the final representation. This makes a good amount deal of sense, though there are some curiosities, like *kicks* and *the* being paired into a single constituent.

4.2 Pre-fix Arithmetic

Since our experiments on SNLI with natural language yielded evidence of a sensible parsing strategy, we wanted to more rigorously test our model's ability to learn to parse. Natural language is ambiguous, and as we showed in Figure **??**, there can be multiple valid parses for a single sentence. This makes it difficult to interpret whether a model is able to succeed at the task of parsing alone. To probe this question of the model's ability to parse, we built a pre-fix arithmetic dataset, referred to as Listops.

Unlike in natural language, Listops us constructed in such a way that any given string has only a single valid parse. Furthermore, we carefully tune the difficulty of the dataset so as to ensure that a model which does supervised parsing, like SPINN (Bowman et al., 2016), is able to succeed at the dataset, while a model that does not consider parsing structure fails. Therefore, to build Listops, we test it on SPINN and a standard bi-directional RNN with Gated Recurrent Units (GRU;

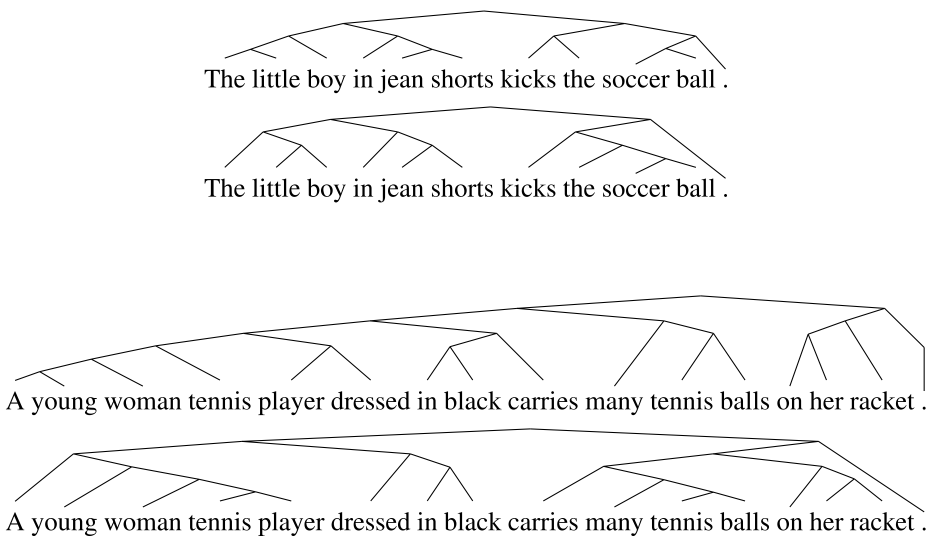
The little boy in jean shorts kicks the soccer ball .
The little boy in jean shorts kicks the soccer ball .
A young woman tennis player dressed in black carries many tennis balls on her racket .
A young woman tennis player dressed in black carries many tennis balls on her racket .

Figure 3: Two example of sentence parses generated by our ST-Gumbel Chart-Parsing model on the top, and the true parse from the Stanford parser at the bottom.

Cho et al., 2014). We then train our ST-Gumbel Chart-Parsing model on the resulting dataset. We choose our ST-Gumbel variant because it illustrated better performance on natural language sentence classification.

For all models trained on Listops, we use a 128 hidden dimension size. The RNN, SPINN, and Choi ST-Gumbel models are substantially tuned to ensure that their reported performances are accurate. Our ST-Gumbel Chart-Parsing model is slow to train, and therefore is not heavily tuned.

Results on the Listops dataset are shown in Table 3. We see that SPINN succeeds at the dataset, achieving an accuracy of 91.3%, while a standard RNN fails with an accuracy of 72.2%. Interesting, the Choi et al. (2017) model performs as poorly as the RNN, with accuracy plateauing at 72%. Our model however achieves an accuracy in between SPINN and RNN, at 84.5%! Clearly then, our model is learning to parse. It doesn't fully solve the dataset, and falls short of SPINN, but it significantly outperforms RNN and the Choi et al. model.

Model	**Listops Acc.**
128D RNN	72.2
128D SPINN	**91.3**
128D ST-Gumbel (Choi)	72.0
128D Chart-Parsing (ours)	<u>84.5</u>

Table 3: Accuracy on the Listops dataset, which is composed of sequences of pre-fix arithmetic.

These results indicate that while our model isn't able to parse as well as a supervised model like SPINN, it capable of learning to parse unliek existing latent tree learning models. It was able to discover the one true parsing strategy for Listops, which a model building random trees like Choi et al.'s is incapable of doing.

5 Conclusion

Discussion From the quality of parses our model produces in natural language, and from its performance on Listops, we can conclude with certainty that our model is capable of learning to parse. It falls short of supervised methods but it substantially outperforms current existing latent tree learning models.

However, since the model's performance on sentence classification with SNLI is significantly lower than Choi et al. 2017, we need to enable our model to learn to parse on natural language more efficiently. The good performance on Listops shows that our model is capable of finding a correct, useful, parsing strategy. However, other aspects of the model can be improved to facilitate better gradient flow to the parser, and to more effectively use the chosen parse.

Future work We propose to strengthen the composition function of the Tree-LSTM (Equations (1)–(3)). We can do this in a few different ways. Motivated by the impressive results of using max-pooling with bi-directaionl RNNs (Conneau et al., 2017; Nangia et al., 2017), we will test using max-pooling with Tree-LSTMs. We will also test the use of highway layers inside the Tree-LSTM to enable stronger compositions.

Additionally, we propose testing scheduled temperature annealing with our Gumbel-Softmax Chart-Parsing. This could enable the model to take advantages of the ST-Gumbel approach, while still allowing the model to better explore the parameter space by initially building soft trees.

Finally, we plane to build a new model architectures that addresses the issue of slow training time, while still retaining the model's parsing ability. While the model proposed in this model is able to parse, and shows potential on natural language, its slow training time makes it impractical. By leveraging the architectural lessons learnt with this model, we believe we can build a computationally faster model capable of the same parsing quality.

Acknowledgments

A massive thank you to Michael Gill, the Capstone course advisor and instructor for this project. His insightful comments have been helpful in guiding the direction of the project.

References

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

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)*. Association for Computational Linguistics, Lisbon, Portugal, 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 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Berlin, Germany.

Kyunghyun Cho, Bart van Merrïenboer, 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)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734.

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.

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

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (to appear)*.

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.

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

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

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. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tadao Kasami. 1965. *An efficient recognition and syntax analysis algorithm for context-free languages*. Air Force Cambridge Research Laboratory, Bedford, MA.

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.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R. Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. In *Proceedings of RepEval 2017: The Second Workshop on Evaluating Vector Space Representations for NLP (to appear)*. Association for Computational Linguistics.

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

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.

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.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks.

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

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

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)*.

Daniel H. Younger. 1967. *Recognition and parsing of context-free languages in time n3*. Information and Control.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures.