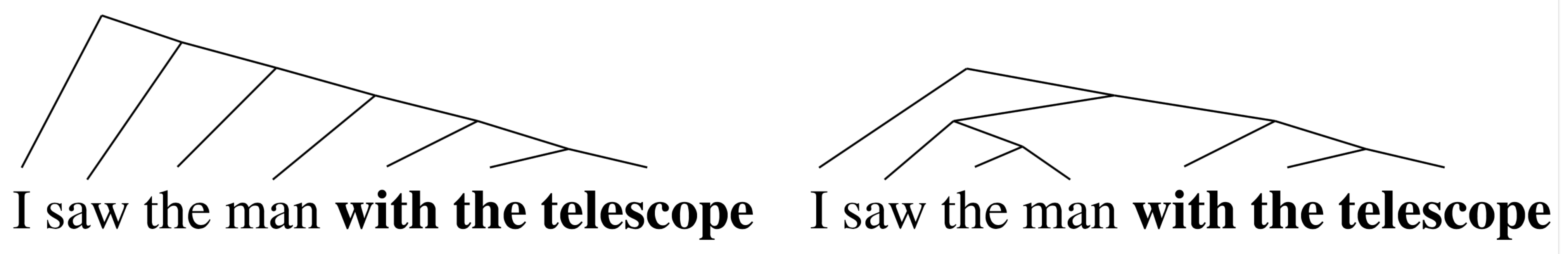
Chart Parsing for Latent Tree Learning

Nikita Nangia, Samuel Bowman, Michael Gill
CDS Capstone Project


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

Introduction

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 natural language inference (NLI). In spite these results, analysis of the parses from these models has shown that they do not learn a consistent, informative strategy, but simply build shallow, random trees. They 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. The model in this paper uses a method inspired by CYK-style (Cocke, 1969, Kasami, 1965, Younger, 1967) chart parser to learn to parse while using the produced parses to build sentence representations. The architecture is inspired by both Maillard et al. (2017) and Choi et al. (2017).

Model

Following the chart parses method this model computes every valid composition of constituents (Figure **??**), i.e. tokens and subphrases, using a TreeLSTM (Hochreiter and Schmidhuber, 1997). When a constituent can be built up in more than one way, then we compute a score for each possible composition by taking a dot product of each hidden state h_i with a scoring vector v,

$$e_i = h_i \cdot v$$

for all possible compositions $i \in [1, n]$. The scoring vector is randomly initialized with all other parameters and has the same hidden size as the hidden states of the TreeLSTM, we use 300-D hidden states.

After getting a score for each composition, we compute a normalized scalar weight using Gumbel-Softmax (Jang et al., 2016),

$$w_i = \frac{\exp((\log(e_i) + g_i)/\tau)}{\sum_{k=1}^{n} \exp((\log(e_k) + g_k)/\tau)},$$

where $g_1, \ldots, g_n$ Gumbel$(0,1)^2$ and τ is a temperature parameter.

We then either use the computed weights to calculate a linear combination of compositions or we use the Straight-Through (ST) Gumbel variant of the estimator and select the composition with the largest weight.

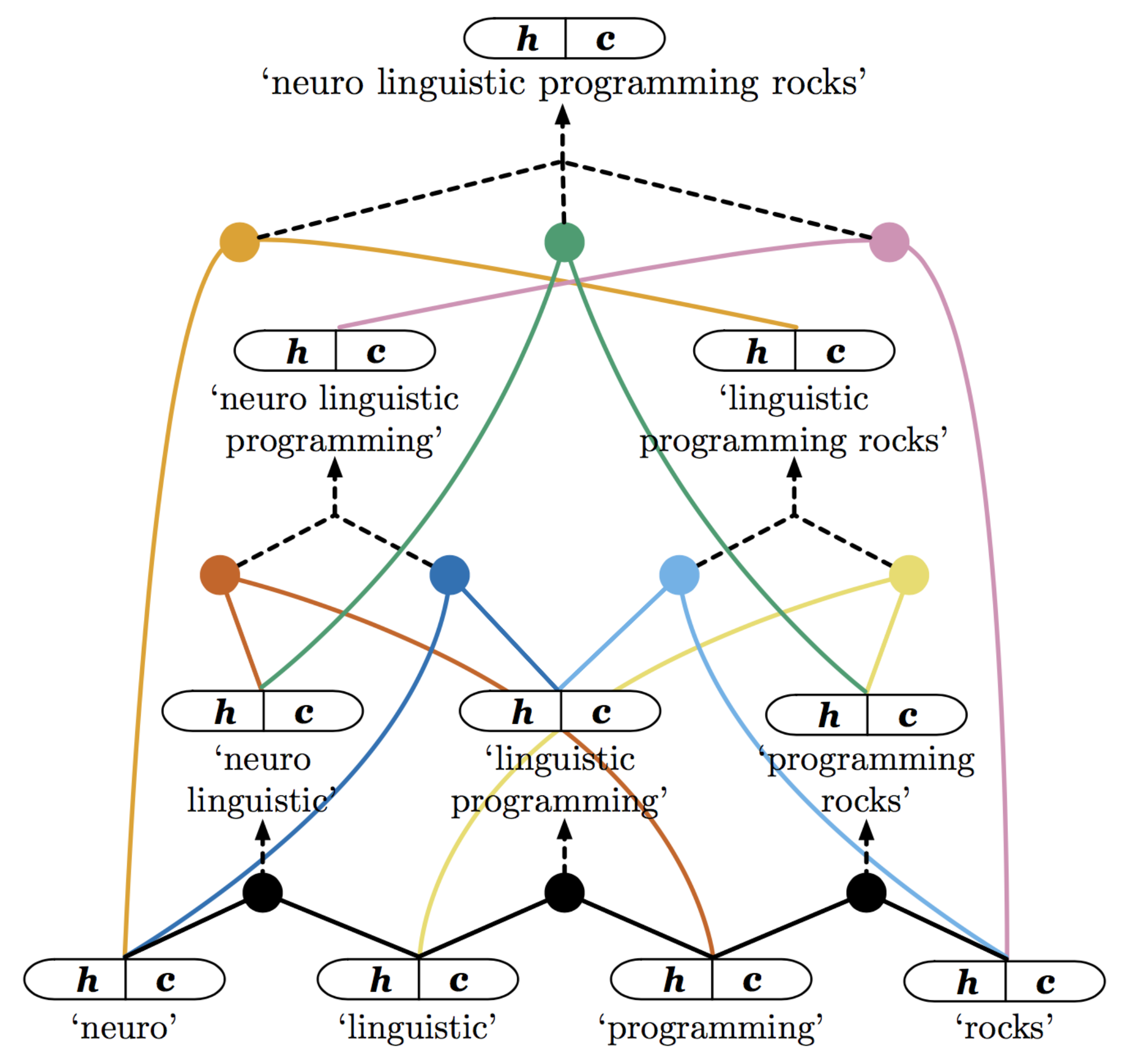


Figure 1: Model architecture when making soft decision at the combination stage. This figure is borrowed from Maillard et al. (2017).

Experiments on Natural Language

To test the efficacy of the model, we choose the task of Natural Language Inference (NLI) and use the Stanford Natural Language Inference (SNLI) corpus. In Figure 2 we show some examples of parse trees chosen by our model on the SNLI dev-set.

Accuracy We build and train three versions of our model. The first does soft combinations of compositions 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 assigned weights to the compositions before doing a linear combination. Table 1 shows accuracy results for all models and baseline models. For baseline comparison, we use the SPINN model (Bowman et al., 2016) which does supervised learning of the parser, the Choi et al. (2017), and the Maillard et al. (2017) model that also uses a chart-parser for unsupervised parsing.

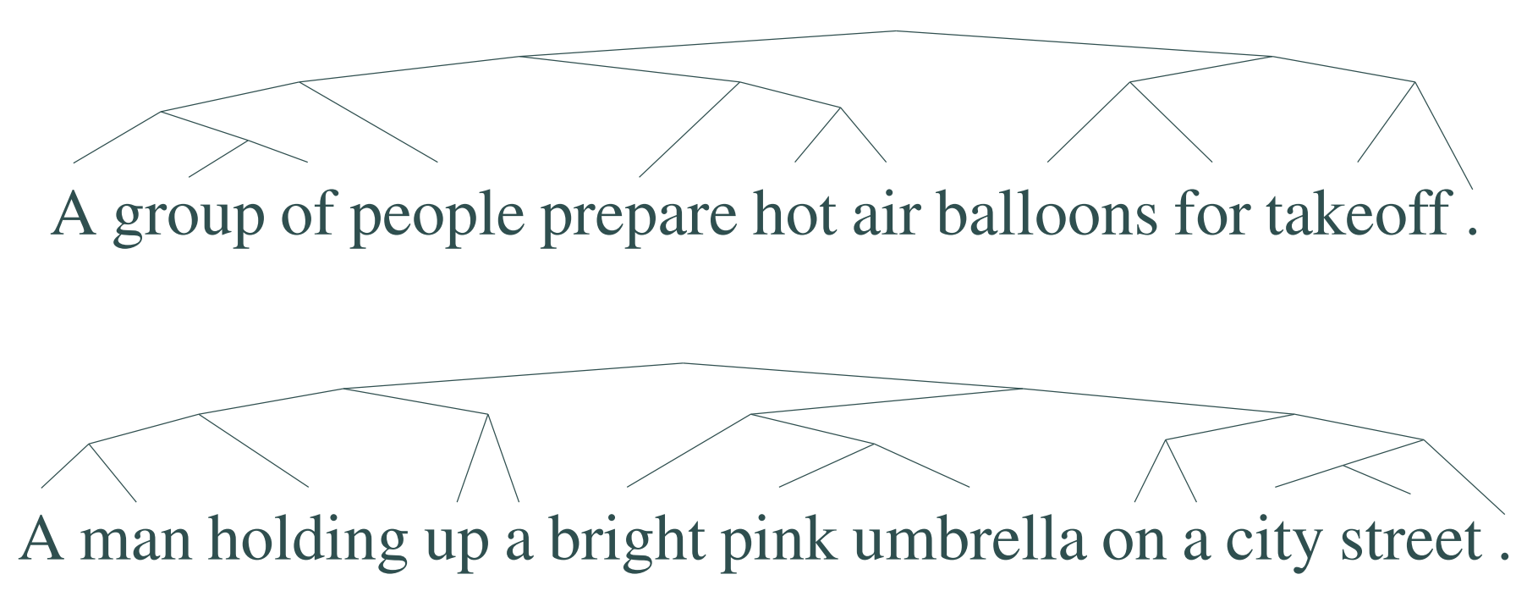

A group of people prepare hot air balloons for takeoff .
A man holding up a bright pink umbrella on a city street .

Figure 2: Examples of trees from our ST-Gumbel Chart Parsing model.

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.

F1 with Ground Truth While our model does not outperform the Choi et al. (2017) model, it does have better performance on metrics that evaluate the trees chosen by the model. It was shown in Williams et al. (2017) that the Choi et al. (2017) model settles on random, shallow trees. Our model seems to find a slightly more sensible parsing strategy. In table 2 we compare the trees chosen by our model to ground truth trees. We see that the ST-Gumbel CP model significantly outperforms the random-trees model.

	Model Type		
F1 wrt.	**Random**	**ST-Gumbel**	**Gumbel**
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: F1 score against left branching, right branching, and ground truth trees on SNLI test set.

Grammar To look at parsing quality of the models, we compute the accuracy of the model on intermediate node types like adjective phrases (ADJP), noun phrases (NP), and prepositional phrases (PP). We see that both ST-Gumbel and Gumbel CP models perform substantially better than random trees on most node types, indicating tremors of sensible parsing choices.

	Model Type		
Node Type	**Random**	**ST-Gumbel**	**Gumbel**
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**

Table 3: Accuracy on correctly composing various node types on SNLI test set.

Qualitative Figure 2 shows examples of trees chosen by the ST-Gumbel CP model. These trees a difficult to qualitatively assess since they do not closely resemble standard PTB grammar. We do find that the average tree depth on SNLI (Table 1) is close to the ground truth average of 4.39. So the trees aren't as shallow as Choi et al. (2017) trees while is encouraging but not very conclusive since SNLI has shorter sentences.

Experiments on Pre-Fix Arithmetic

Since natural language parses are difficult to interpret, we test the model's performance on an artificial dataset wherein a model needs to find the single correct parsing strategy to succeed at the task. We built a pre-fix arithmetic dataset called Listops. Table 4 shows accuracy results on this dataset.

Model	Listops Acc.
128D RNN	72.2
128D SPINN	**91.3**
128D ST-Gumbel (Choi)	72.0
128D Chart-Parsing (ours)	<u>80.5</u>

Table 4: Accuracy on the Listops dataset which is composed of sequences of pre-fix arithmetic.

The Choi et al. (2017) model achieves roughly the same accuracy as the RNN; our ST-Gumbel CP model however, achieves an accuracy in between SPINN and RNN. This curious result indicates that the CP model is finding a parsing strategy that is more sensible than Choi et. al.'s model (2017), though it is far from perfect.

Discussion

We've seen that our CP model, particularly when it makes hard composition decisions with ST-Gumbel, settles on a parsing strategy that is decidedly better than random trees, which is an improvement on the Choi et al. (2017) model. Since the model's performance on the pre-fix arithmetic dataset is in between that of a SPINN and RNN model, there is compelling reason to pursue hill-climbing strategies to improve the model. There are a few directions for future work,

- Improve composition function. Leverage max-pooling trick from standard RNNs in TreeRNNs.
- Test temperature annealing to combine possible advantages of both soft and hard decisions in the CP model.
- Instead of building of exploring all possible parses with a chart-parser, select a subset of possible parses. This would considerably cut down on training time.

These methods, particularly improving the composition function, could help imrpove the signal to the parser and encourage the model to learn a better parsing strategy.