Assignment 4 Report

Tomer Gill  
318459450  
gilltom

**The paper** I chose to implement is "**Learning to Compose Task-Specific Tree Structures**" by Jihun Choi, Kang Min Yoo & Sang-goo Lee that was published in November 2017.

Link to the paper: <https://arxiv.org/pdf/1707.02786.pdf>

# The Paper's Method

The paper describes a model that learns how to represent a sentence using tree LSTM for a specific task (in our case, SNLI). Because of this I will first describe the general sentence representation method and then I'll describe the SNLI predictions.

## Tree LSTM

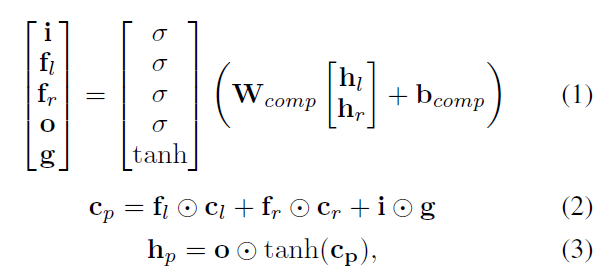
The model is based as an Tree LSTM – The LSTM part means for each represented node (which can be one or more words with a single [h, c] representation. Will be described later) we will have both outer and inner representation which are 2 vectors: **h** and **c** (respectively), which are like the "y"-s and "s"-s of a LSTM RNN. The tree part means we will build our representation of the whole sentence layer-by-layer from the bottom up – at each step the model will choose adjacent nodes from the layer and unite them to a single node (with a single [h, c] representation).

### Leaf Representation

At the beginning, each word is given its own [h, c] representation by encoding it using a linear layer, or alternatively the paper suggests using a LSTM RNN to give each representation info about the previous nodes. This can also be upgraded to a BiLSTM.  
The represented words are the leaf nodes of out tree LSTM.

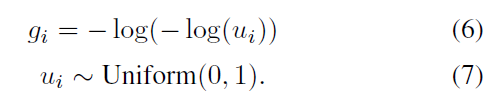
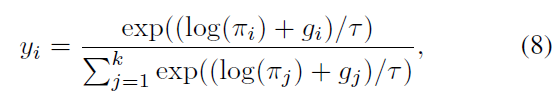
### Parents Representation

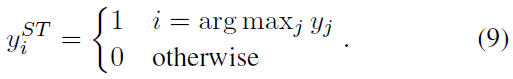
At each step, the model will represent all the possible parents of 2 adjacent nodes. This is done by concatenating the **h** representations of the left node on top of the right node's **h**, then going through a linear layer and different activation functions, we will get 5 gate vectors (as a LSTM does) and with them we will create the [**h**p, **c**p] representation of the parent node:



### Gumbel Softmax

Inspired by the Gumbel Max trick, the Gumbel Softmax uses the differentiable soft max function. The GS "adds" a *Gumbel Noise* a set of unnormalized probabilities π1, · · · , πk, a sample yk, …, yk will be:

And the Straight-Through version will be: 

### Parents Selection

To choose which nodes will be united to a single node, each possible parent is scored: using a trainable *query vector* is dot-multiplied with each possible parent. Then those scores are going through a softmax function.

In the testing phase the parent with the maximal score is chosen, but in the training phase the scores are going through the Gumbel Softmax function thus giving us the 2 samples: **y** and the one-hot version **yST**. In the forward pass **yST** is used for fast computing, but in the backward pass to update the parameters **y** is used. In DyNet this can be done like this:

dy.nobackprop**(**y\_st **-** y**) +** y