# Exercise 3 Report – Penn Treebank Language Model

## Model Architecture

The model we used is the model from “Element-Research”. We used a standard LSTM network with the following layers:

1. An embedding layer (Lookup Table), which encodes each one of the 10,000 words in the vocabulary to a real-valued vector of size 132.
2. The next layer received as input sequences of 20 encoded words (this value proved to be good empirically) which are fed into a 2-layer LSTM, with a hidden state of size 132, and an output of the same size.
3. A SoftMax layer the size of the vocabulary (10,000), in order to obtain a prediction score for the next word in the sequence. In total, our model has **2,978,368** parameters. Our best model achieved perplexity of **113.042** on the test set, and 117.56 on the validation set.

## Data Augmentation

We didn’t use any data augmentation.

## Initialization

We used a uniform initialization between -1 and 1 for the LSTM layers, and used the default initializations for the LookupTable and Softmax layers.

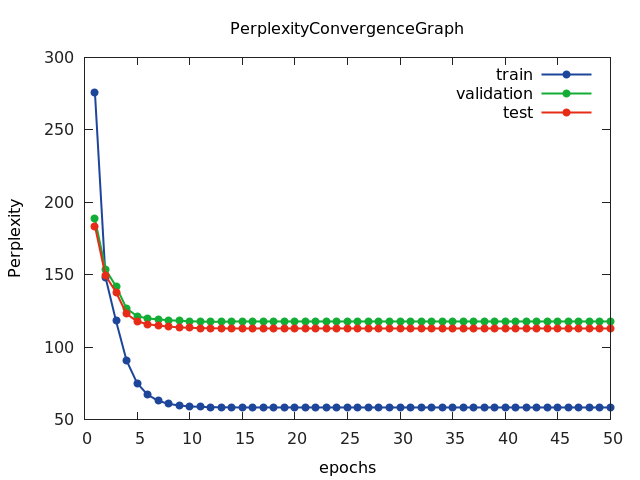
## Optimization

* We used the ClassNLLCriterion, which is recommended for multiclass classification problems. Each word in the vocabulary is considered as a different class.
* The optimization method we used in SGD with momentum (0.9), and we employed a decaying learning rate. We tried different learning rates until we reached the following schedule:
  + Initial learning rate is 1.
  + Keep learning at rate 1 up to (including) epoch 3.
  + Halve the learning rate after each epoch starting from the 4th.
  + Stop decreasing the learning rate when you reach 0.0001

## Things we considered

1. We saw from the beginning that the LSTM model overfits to the training set, so we used the learning rate decay method which succeeded in achieving a reasonable test perplexity.
2. In order to avoid exploding gradient problems, we clipped the gradients during training such that the L2 norm of the vector containing all of the gradient will be at most 5.
3. We saw that different length of sequences at different batch sizes have a noticeable effect on the performance of the model. Setting the batch size and the sequences length to 20 (as one of the examples in Element-Research’s implementation) gave good results.

## Results



## Generated Sentences

In order to generate sentences from our trained language model, we first “warmed-up” our model with the prefix “buy low sell high in the”. After feeding in the last word, we generated the rest of the sentence word-by-word, by drawing a word with a probability equal to that word’s score by the language model. We kept drawing words until the special token ‘<eos>’ was drawn. Here are 5 example sentences:

1. buy low sell high in the secondary market
2. buy low sell high in the portfolio he said
3. buy low sell high in the past couple of years to her life
4. buy low sell high in the 20th held range of a combined loss of five cents a share
5. buy low sell high in the deterioration of next year