M.Sc Progress Report

Neuro-Linguistic Prediction with Penn Treebank (PTB)

Name: Omer Luxembourg *Advisor*: Prof. Haim Permuter

1 Introduction

Natural language processing has gone trough significant progress in the last years, by developing models in neural networks. I conduct experiment on the Penn Treebank (PTB) which is a well known natural language problem. The goal was to understand what regularization methods and model structure will improve the generalizing learning process. In the following sections, I will present the baseline model and state-of-the-art model results and conclusions.

2 Model Implementation

2.1 Baseline Model

Using Recurrent Neural Network (RNN) in natural language processing is very important. The reason is that natural language is highly context-dependent, thus reusing the output as an input in RNN models will help the network understand how to generalize the language.

The first baseline model, in figure 1, is constructed with 2 layers of Long Short-Term Memory (LSTM) cells which is a RNN implementation that does not has the problem of vanishing or exploding gradients (which are crucial for the learning process). In addition to the LSTM layers, exists embedding layer in the input and output, which will encode and decode the words to a fewer neurons than the vocabulary size - this means instead of inserting the first LSTM layer with 10,000 neurons (as of the vocabulary size of PTB data), we have a smaller input size by 10 to 100 (200-600 neurons). For this baseline model, extra dropout layers were added in between each layer mentioned before, with rate of 50

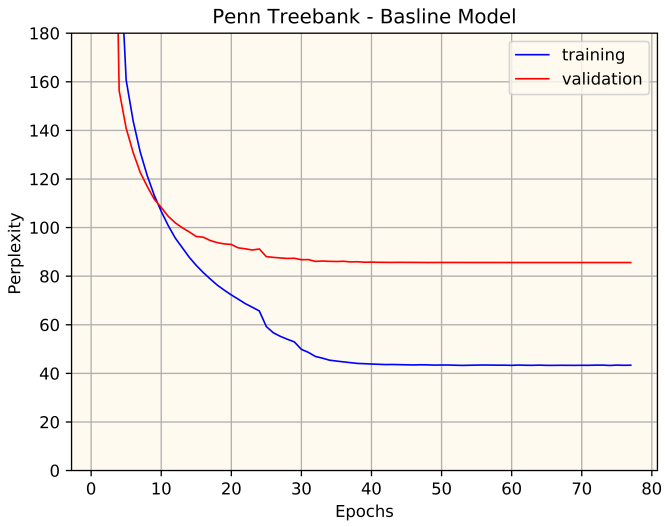

Figure 1: Baseline Model Perplexities

The optimizer of the model is the basic Stochastic Gradient Decent (SGD) with momentum of rate 0.9. The starting learning rate is 1.1 and it will decay by 2 every time that the valid perplexity is getting worsen. The embedding layer is at the same size of the hidden layer of the first (and also second) LSTM cell - which is 650 units. The model been trained for 80 epochs, each epoch has batch size of 20, which is constructed of input sequence length of 35 words. Early stopping is activated and will prevent further train of the last 15 no-improving epochs. As you can see in figure 1, the validation samples perplexity is 85.60 bits after the train session.

2.2 Advanced Model

As for the advanced model, in figure 2, the same baseline model been improved with additional LSTM's variational dropout mask [1]. For the baseline model we used a native dropout, which will affect LSTM's inputs and outputs alone, and will change for every time step. The variational dropout is an implementation of dropout in the inputs layer, hidden layer and outputs layer. In addition we use the same mask of the dropouts for every time step, which will prevent the recurrent connection's dropout mask to worsen the results.

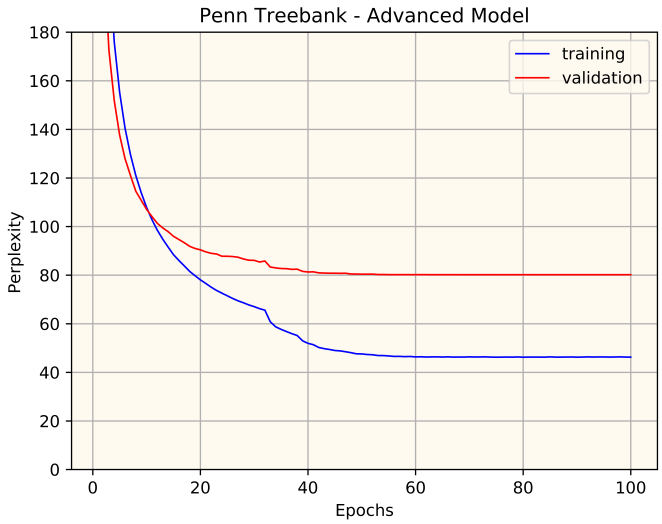

Figure 2: Advanced Model Perplexities

The model is implemented by Tensorflow, thus running recurrent (variational) dropout will lead to train on the CPU instead of using NVIDIA cuDNN library, which meant a slower train time. The model is as the baseline model (in figure 1) and differ only in the new dropout implementation. It has been trained for 100 epochs.

2.3 State-of-the-art Model

The last implementation is state-of-the-art. The model is built according to three main concepts: AWD-LSTM, MoS and dynamic evaluation.

- AWD-LSTM stands for Averaged SGD (a.k.a ASGD) Weight-Dropped (a.k.a drop connect) LSTM. It is one of the most popular today's language models. The purpose of ASGD is to calculate the new weights of the model by averaging all weights that have been learned since a

trigger point which will be chosen in the training session on run time. Weight-dropped LSTM is meant to do the same as the variational dropout in LSTM, but instead of dropping-out the neurons (as for variational), the weights are the one thats been dropped, meaning instead discarding columns in the weight matrix, we discard elements in the same matrix [2].

- MoS stands for Mixture of Softmax. The idea is that a single softmax layer at the output of the model (at the vocabulary size) won't be good enough to represent the dimensions of all contextual outputs. Thus using MoS is giving a context-dependent distributions a higher rank to be expressed in the model's outputs [3].
- Dynamic evaluation is a method of adapting the model parameters that have been learned through training time, to samples that are given through evaluation time. The concept is to provide a better model then using a model with fixed parameters that is trying to capture reoccurring patterns [4].

The model is implemented in Pytorch 1.2, and been trained, fine-tuned and dynamic evaluated. See figure 3.

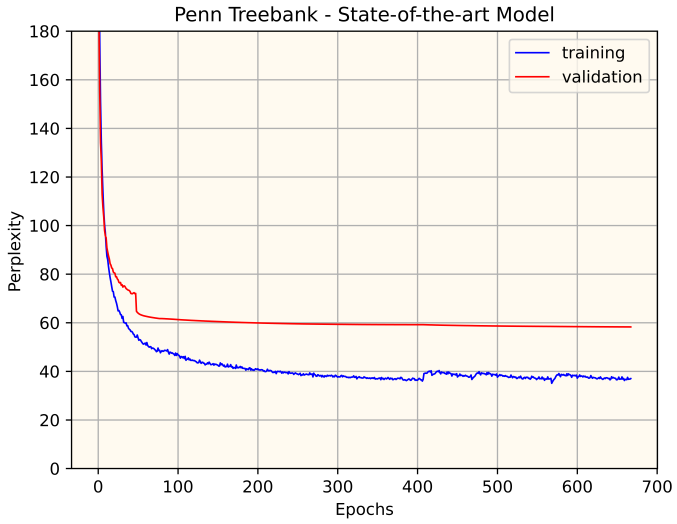

Figure 3: State-of-the-art Model Perplexities (without finetuning and dynamic evaluation)

3 Results

The results of all the models mentioned above can be seen in Table 1.

4 Future Work

In future work we will research how to use new regularization techniques to improve context-dependent data-sets, which includes the Penn Treebank and also will try to be implemented on stocks predictions problems.

Model	Parameters	Train	Validation	Test
Baseline LSTM	19.7M	43.35	85.60	81.93
Variational LSTM	19.7M	46.25	80.17	76.29
AWD-LSTM + MoS w/o finetune	21.5M	37.01	58.27	56.08
AWD-LSTM + MoS	21.5M	39.97	**57.71**	**55.51**
AWD-LSTM + MoS + Dynamic Eval	21.5M	-	**49.86**	**49.12**

Table 1: Basline, advanced and state-of-the-art model perplexities for Penn Treebank database

References

[1] Y. Gal and Z. Gharamani, "A theoretically grounded application of dropout in recurrent neural networks," *in NIPS*, 2016.

[2] N. S. S. Merity and R. Socher, "Regularizing and optimizing LSTM language models," *ICLR*, in 2018a.

[3] Z. Yang and Z. Dai, "Breaking the softmax bottleneck: a high-rank RNN language model," *in ICLR*, 2018.

[4] B. Krause and E. Kahembwe, "Dynamic evaluation of neural sequence models," 2017.