NLU project exercise lab: 9

Mateo Rodriguez (239076)

University of Trento

md.rodriguezromero@studenti.unitn.it

For the first part of the lab four variations of a model were trained and tested, each variation of the model resulted in an improvement of the perplexity. For the second part of the lab, the best performing model from the first part was taken and tested with three more variations, none of them improved results.

1. Introduction

- For the first part of the lab, there are two main models, one that uses RNN and the other LSTM
- For the second part the extra modifications added to best model from the last part were: weight tying, variational dropout and non-monotonically triggered AvSGD

To maintain the variables as similar as possible the loss used for all the models was cross entropy and a batch size of 32, maxing out available computing power. Since the optimizers differ the learning rate cannot be held constant.

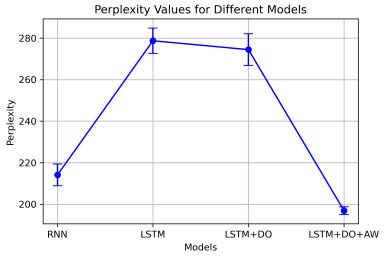
2. Implementation details

Going more in detail, we utilized the Penn Treebank dataset, and the 4 model variations tested for part 1 were the following: RNN, LSTM, LSTM plus dropout, and LSTM plus dropout with AdamW. The models were trained all using a hidden and embedding size of 256, and with SGD except for the last one.

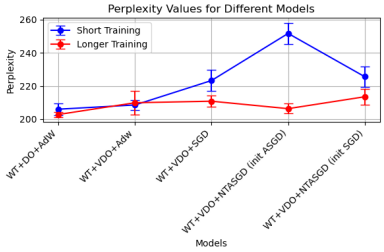
Regarding the second part of the lab, the best performing model was of course the last configuration, so it was used for the this part to add the modifications mentioned previously in an accumulative way. For the first modification, weight tying, it was only necessary to specify that the weights of the output linear layer were the same as the weights of the embedding layer, this has been shown helpful in cases where the input has a similar structure as the output. The next modification was using variational dropout, which involves simply keeping the same dropout mask for both input and output of the LSTM. This can also be used in the between layers of the LSTM, but since the tested models only used one, the dropout was only applied to the input and output of the LSTM. For the final modifications two models where tested. Both have weight tying and variational dropout in them but what differs is the optimizer. Up until here the models used AdamW, so to have a just comparison one run was trained using SGD and the other NT-ASGD. The implementation of NT-ASGD was a little tricky, pytorch by default only includes ASGD, and the only implementation available online didn't seem to work [1]. So the resulting optimizer includes the conditions mentioned in the pseudocode of the paper and used in the source-code [2], into the training loop and manually change the parameters based on the conditions. The resulting optimizer worked slightly better than the standard SGD.

3. Results

For the first part of the lab, the lowest perplexity was obtained by the last model configuration, the LSTM model with dropout and AdamW, but what's interesting is that the second best performance was achieved by the RNN network



For the second part of the lab, results were the following:



Here it's clear that the variations tested were not beneficial for the performance of the models, in fact the best model obtained out of all is the best model from part 1. The models were trained twice because an initial training of 100 maximum epochs and patience of 25 led to unsatisfactory results (blue line). For the second training (red line) the models had a cap of 500 epochs and patience of 100 allowing a longer overall training, and reaching much better results.

The two NT-ASGD variations shown in the figure arise from the fact that in the source code ASGD is used as the base optimizer from which on top the NT variations are added. The graphs show that the ASGD initialized version requires longer training to perform better compared to the SGD initialized version.

In general results meet expectations, specially confirming the superiority of AdamW over SGD, not only in performance but also in stability of the results. Regarding the other modifications like weight tying and variational dropout, it's possible that they are beneficial for larger models and not ones using single layer LSTMs. It's also important to note that these results are based of training in a single dataset, meaning for a full analysis of the effect of the modifications added to the models it'd be necessary to run similar testing on different datasets and tasks. Compared to the paper the authors use a batch size of 80

and 8000 epochs which are both much higher than the settings tested.

4. References

[1] A. U. Durmus, "Awd-lstm pytorch implementation," https://github.com/ahmetumutdurmus/awd-lstm, Year.

[2] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and Optimizing LSTM Language Models," *arXiv preprint arXiv:1708.02182*, 2017.