NLU course projects lab 4: LM

Nicola Muraro (248449)

University of Trento

nicola.muraro@studenti.unitn.it

1. Introduction

The purpose of this assignment is to implement a language model based on neural architectures. Specifically, the task of this model is to predict the next token given a set of tokens, referred to as the context. To achieve this goal, I used an LSTM on which several modifications were made to improve its performance. In the first part of the assignment, various optimisers were analysed alongside the use of dropout. In the second part, instead I implemented more sophisticated regularisation techniques described in the article written by Merity et al. [1]. This report briefly summarises the studies conducted and the results.

2. Implementation details

The first part of this assignment focuses on studying LSTM as the foundation of our architecture, followed by two incremental modifications: adding dropout layers and using AdamW.
I started by replacing the original architecture (based on RNN) with LSTM.
Next, I made the first modification by adding dropout layers: one after the embedding layer and another before the output layer. Finally, I implemented the last required modification by replacing SGD with AdamW. Since changing the optimiser also requires adjusting the learning rate, I made the necessary adjustments to the learning rate to ensure an optimal performance.

The second part involves starting from the initial network (LSTM with SGD) and incrementally applying regularisation techniques outlined by Merity et al. [1].
First, I implemented weight tying between the embedding layer and the output layer. This technique allows multiple layers in the network to share the same weights, aiming to regularise the entire architecture. To enable weight sharing, the layers involved must have the same dimensions, so I adjusted their sizes to ensure compatibility.
The second modification involved using a variation of the classic dropout technique. Traditional dropout applies a different mask at each time step. In contrast, this variant, known as "Variational Dropout", applies the same dropout mask at each time step.
Finally, I implemented the last modification: using Non-monotonically Triggered AvSGD. This is a variant of AvSGD where the switch to AvSGD is based on the network's state during training. Specifically, if SGD fails to improve the solution over several consecutive epochs, it is replaced with AvSGD in the hope of further optimizing the solution. Since the timing of this switch is not predetermined, this technique is expected to provide additional performance improvements compared to both standard SGD and AvSGD.

3. Results

The results used to evaluate the effectiveness of the implemented techniques were obtained using the evaluation set. This approach ensures better generalisation. Using the evaluation set to fine-tune hyper-parameters and find an optimal configuration allows for more reliable results on the test set. In general, the best machine learning models have high capacity and are properly regularised. Since many of the previously applied techniques are forms of regularisation, I started with a relatively large model. For all configurations, the batch size for the training segment was kept constant at 64, while for the evaluation it was set at 128.

In the first part, the initial model was created with the following specifications: $emb\_size = 600, hidden\_size = 500, lr = 5$. In this case, I also used a scheduler, specifically the 'LinearLR' variant. This approach allows for a higher learning rate in the initial phase of training, while gradually reducing it to a more conservative value toward the end, as the model approaches the optimum.
Then, the use of dropout further improved the results. In this case, the dropout probability was set to $p = 0.5$.
Finally, after replacing the optimizer with AdamW, I adjusted the learning rate to a more conservative value $lr = 0.0005$.
As we can see from the results shown in Table 1, all the modifications led to improved performance, surpassing also the baseline. The baseline refers to the original network from which we started our modifications (the RNN provided in class), but with the learning rate and embedding size set to the same values specified above for the LSTM network.
Also, the training evolution of the best model are depicted in Figure 1.

Model	Eval PPL	Test PPL
BASELINE (RNN)	900.31	878.2
LSTM	139.33	136.33
LSTM with dropout	130.95	126.31
LSTM with dropout, AdamW	117.85	106.87

Table 1: *Perplexity values for the first part of the assignment*

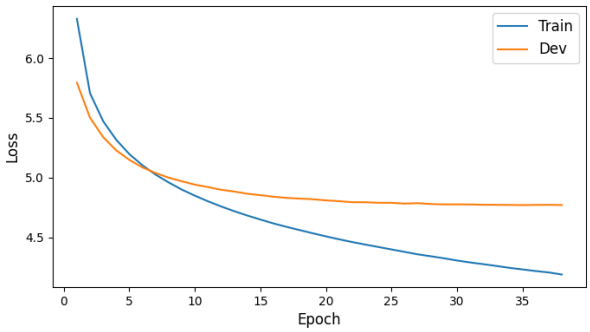

Figure 1: *Trainig loss for the best model of the first part*

For the second part, starting with the same configuration

as before ($emb\_size = 600, hidden\_size = 500, lr = 5, scheduler = LinearLR$), I modified the layer sizes to use weight tying. The model now has $emb\_size = hid\_size = 600$.
Adding Variational Dropout significantly improved the results. In this case, the mask used had a dropout probability of $p = 0.5$. Finally, as shown in table 2, I completed the assignment using NT-AvSGD. The last model switches to AvSGD only in the last few epochs, yet this still enables the model to reach a lower perplexity value. In Figure 2, the final model's training progression is represented. All these modifications led to better results than the baseline for this part of the assignment (the first LSTM network developed in the previous section).

Tecnique	**Eval PPL**	**Test PPL**
BASELINE (LSTM)	139.33	136.33
W.Tying	124.37	120.51
W.Tying, Var.Drop.	94.62	92.16
W.Tying, Var.Drop., NT-AvSGD	91.26	88.97

Table 2: *Perplexity values for the second part of the assignment*

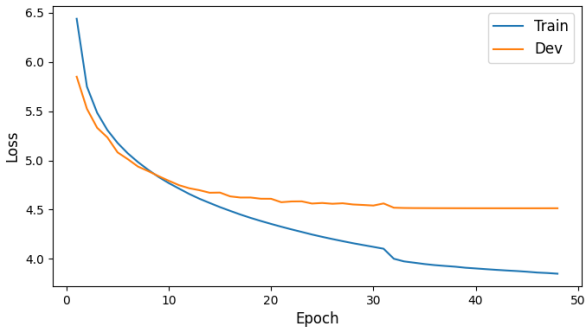

Figure 2: *Trainig loss for the best model of the second part. The dip around 32 epochs is due to the switch to AvSGD.*

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.