NLU course project - LM assignment

Pietro Bologna (248715)

University of Trento

pietro.bologna@studenti.unitn.it

1. Introduction

The LM assignment begins with the implementation of an LSTM architecture and progressively increase its performance with various techniques. In the first part, I replaced the RNN with a Long-Short Term Memory (LSTM) network; then I added two dropout layer: one after the embedding layer and the other one before the last linear layer. Next, I substituted the Stochastic Gradient Descent (SGD) with AdamW. In the second part, I applied the Weight Tying, the Variational Dropout (no DropConnect) and the Non-monotonically Triggered AvSGD to the LSTM model. The objective throughout these steps is to always keep the perplexity (PPL) below 250.

All the techniques used in this assignment are described in the paper '*Regularizing and optimising LSTM language models*' by Stephen Merity [1].

2. Implementation details

First of all, I implemented the **LSTM**[2] and added the two **dropout layers**[3] (one after the embedding layer and the other before the linear layer). When adding a dropout layer we have to set a parameter called 'probability'. This value *p* indicates the probability of each element of the input tensor being set to zero during training. After various tests, I set a probability of *p=0.1*, which means that each element has a probability of 10% of being zeroed. With the introduction of dropout, we increase the generalization capacity of the model by reducing the possibility of overfitting and this results in a more robust model. After that, I replaced the Stochastic Gradient Descent with the **AdamW**, which is an optimization algorithm that adjusts the learning rate for each parameter individually, allowing it to adapt the learning rate throughout training.

In the second part of the assignment, I applied **weight tying**, a regularisation technique that makes it possible to considerably reduce the number of model parameters by sharing the weights between the embedding and the output layer. To do so, I shared the weights between the two layers. Then, I replaced the standard dropout with the **variational dropout**[4] which is a technique that applies the same dropout mask across all timesteps in an LSTM. Finally, I implemented the **non-monotonically triggered AvSGD**. In this last part, the model starts using the SGD and the code checks if certain conditions are met to switch from SGD to ASGD. If the development loss has not improved over a certain number of epochs and the current development loss is greater than the minimum loss from previous epochs, the model changes to ASGD. This optimizer averages model parameters over time and this reduces the variance introduced by noisy gradient updates.

3. Results

In the Table 1, I present the best results obtained for each of the parts of the assignment. I performed a fine-tuning process in order to find the best hyperparameters for satisfactory results. After a first phase where I focused on the learning rate, I established this two values according to the optimizer used. For the SGD I set a *lr = 5* and for the AdamW *lr = 0.001*

These results described in the table all refer to the implemented LSTM model:

	Experiment type	PPL
1	SGD	137.03
2	SGD + Drop	128.51
3	AdamW + Drop	120.02
4	SGD + Weight tying	109.24
5	SGD + Drop + Weight tying	104.41
6	SGD + VarDrop + Weight tying	88.28
7	SGD + AvSGD + VarDrop + Weight tying	88.94

Table 1: *Results*

As we can see, in the first experiment we obtain a not very satisfactory result (*PPL = 137.03*) and if we go to analyse it's loss graph (left image of Figure 1) we notice that we are overfitting. Then, if we look to the second result, we notice that by introducing dropout we obtain a better perplexity result. Now, if we look to the third result in the table, we notice that using the AdamW optimizer we get faster convergence and a lower perplexity. With this implementations our model is still overfitting.

For the second part, we notice that after integrating weight tying we obtain a good result because we have reduced the number of parameters and thus improved the generalisation. Then, experiments 5 and 6 show the results obtained using weight tying combined with standard dropout and weight tying with variational dropout, respectively. As we can see, with variational dropout we achieve a better results compared to the other one. As also shown in the right image of Figure 1, in addition to having reduced perplexity, there has also been a reduction in overfitting. Finally, if we compare the last result (obtained with the AvSGD) with the result obtained without it, we see that we do not obtain an improvement, but rather the perplexity value remains fairly stable.

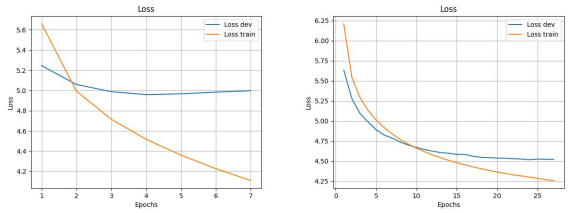

Figure 1: *Losses (overfitting and no overfitting)*

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," in *International Conference on Learning Representations (ICLR)*, 2018.

[2] pyTorch, "Long short-term memory (lstm)," https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html, 2024.

[3] ——, "Dropout," https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html, 2024.

[4] Salesforce, "awd-lstm-lm," https://github.com/salesforce/awd-lstm-lm/blob/master/locked_dropout.py, 2018.