NLU first course project

Carlotta Giacchetta (248321)

University of Trento

carlotta.giacchetta@studenti.unitn.it

1. Introduction

In the context of optimizing recurrent neural networks (RNNs) for Natural Language Understanding (NLU), this report presents an analysis of the modifications made to the existing pipeline. The project focuses on evolving an RNN into a Long Short-Term Memory (LSTM) by adding dropout layers, weight tying, and adopting variants of the classic SGD optimization. Divided into two phases, the work aims to study the incremental impact of these modifications on the model's perplexity, aiming to identify the optimal structure and parameters to enhance performance in the field of NLU.

2. Implementation details

2.1. Firt phase

Throughout the project, significant modifications were made to the existing pipeline to improve the language model's performance. In the first phase, I replaced the RNN with a LSTM network, then I expanded this LSTM with two dropout layers. The first dropout layer was positioned after the embedding layer to prevent overfitting by randomly deactivating neurons during training. Similarly, the second dropout layer was added before the output layer. At the end, I opted for the AdamW optimizer, known for its adaptive learning rate and improved convergence speed, to further optimize training efficiency and model performance.

2.2. Second phase

In the second phase, further functionalities were implemented. Firstly, I implemented weight tying, a technique that promotes parameter sharing between the input and output layers. Following this, I introduced variational dropout, which involves creating a mask of boolean variables to deactivate certain neurons stably during training. [1]

Lastly, I integrated the Non-monotonically Triggered AvSGD optimizer to optimize the model's loss function while considering the historical parameter updates. This involved implementing a control mechanism to evaluate conditions for switching between the regular SGD and AvSGD optimizers. Upon meeting these conditions, a control variable was toggled, signaling the optimizer to utilize the averaged updated values during training. Upon reviewing the ASGD implementation, I identified and addressed three key issues:

1. The ASGD algorithm lacked the utilization of averaged values for parameter updates. To remedy this, I introduced a mechanism to incorporate averaging values ('ax') as parameters of the model.
2. After an examination of the code, I observed that there was no distinction between the model parameters and their corresponding averages. This oversight stemmed from the default setting of 't0' to a large value. Consequently, I adjusted the 't0' value of the optimizer to match 'T' when the trigger condition was met.
3. At the end, an error was identified in the step count of the ASGD optimizer. To rectify this, I ensured that the step count matched the parameter 'k'.

3. Results

To evaluate the implemented models, I performed parameter tuning, focusing mainly on the size of the embedding layer and the hidden layer, the learning rate and patience. Initially, I experimented with low learning rates, but the results obtained were not satisfactory. Subsequently, I gradually increased the learning rate and noticed that up to a value of 10, better results were obtained, but beyond this value, the results began to deteriorate.

Consequently, I decided to increase the overall size of the network, both in terms of width (size of the embedding and hidden layer) and by increasing the patience. Of the different configurations tested, I noticed that the one that produced the best results for the second part of the project was with an embedding and hidden layer size of 600, a learning rate of 10 and a patience of 6. The decision to increase patience was guided by the observation that, using NT-AvSGD as an optimiser, early stopping occurred later, allowing the model to perform better. While, for the first part of the assigment a used a patience of 3. I noticed that with the AdamW optimiser, using a high learning rate leads to disappointing performance. Consequently, I set the learning rate to 0.001 for better results.

Model	lr	ppl
LSTM	10	150.495
LSTM+DROP	l0	140.836
LSTM+DROP+AdamW	0.001	112.588
LSTM+WT	10	120.799
LSTM+WT+VD	10	112.141
LSTM+WT+VD+NT-AvSGD	10	114.403

Table 1: *best results*

All perplexity values obtained were acceptable, remaining below 250. An interesting note is that during the first phase of the project, a decreasing trend in perplexity values is observed as the model is trained. However, when the dropout layers are removed and weight tying is introduced, the perplexity value increases. This suggests that the dropout may be a crucial element of the network. In fact, the addition of the variational dropout led to a significant improvement in the results.

Furthermore, by changing the optimiser and introducing averaging, a switch between SGD and AvSGD occurs at the tenth iteration. Thereafter, the model continues to update the parameters up to the 40th iteration, whereas with the classical optimiser

(SGD), the parameter update stopped at the 14th iteration. This happens because the addition of averaging makes parameter updating much more uniform and smooth.

4. References

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