NLU Course Projects - LM

Ettore Saggiorato (247178)

University of Trento
`ettore.saggiorato@studenti.unitn.it`

1. Introduction

This is the report for the first assignment of the NLU course. The objective of this assignment is to implement and evaluate various techniques designed to enhance model performance in a next-word prediction task. The primary objective is to incrementally modify a baseline RNN model and analyze the impact of these modifications on the performance. Notably, changes may result in either improvements or degradations in the models' effectiveness.

Model performance is assessed using Perplexity (PPL). The experiments are conducted using the Penn Treebank dataset [1], which serves as the training and evaluation benchmark.

2. Implementation Details

This section describes the architectural modifications and techniques implemented to improve the baseline RNN model. Modifications are applied sequentially.

2.1. Architecture

Replacing RNN with LSTM: both architectures are already implemented in Pytorch, switching between the two is trivial and done by simply changing `torch.nn.RNN` to `torch.nn.LSTM`.

2.2. Regularization

Applied to enhance model generalization and robustness:

- **Dropout** is applied in two different locations: after the embedding and before the output layer. Dropout is expected to encourage the network to develop more generalized representations, thereby improving robustness.
- **Weight Tying:** by tying the weights of the input embedding and output projection layers, the total number of model parameters is significantly reduced. This method not only improves computational efficiency but also often leads to better performance in language modeling tasks [2].
- **Variational Dropout:** Variational Dropout [3] applies a consistent dropout mask across the time dimension, meaning the same mask is applied to all time steps within each sample in the batch (i.e., one dropout mask per sample, shared across its sequence length). This ensures that specific parameters are excluded uniformly throughout the sequence, enforcing stronger regularization and potentially enhancing performance.

2.3. Optimizers

Two optimizers are explored as alternatives to standard Stochastic Gradient Descent (SGD):

- **AdamW:** is a more advanced optimizer, it leverages adaptive learning rates and weight decay to achieve faster convergence. Consequently, it enables the model to reach optimal solutions more efficiently.
- **NTAvSGD:** [4] is designed to stabilize SGD, by averaging model weights when validation performance plateaus. It improves resilience against overfitting, especially at higher learning rates.

3. Results

3.1. Architecture

Replacing the RNN with LSTM, as shown in 1 (1.1) mitigates the vanishing gradient problem, achieving lower PPL.

3.2. Regularization

- Dropout alone yielded the most notable improvement by preventing overfitting and enhancing generalization.
- Weight Tying and Variational Dropout further contributed to performance gains relative to the baseline. These methods allowed the model to train for longer durations without triggering the patience.

3.3. Optimizers

- **AdamW:** AdamW exhibited superior convergence speed, achieving lower PPL with fewer training epochs wrt SGD.
- **NTAvSGD:** proved to be more resilient to overfitting, as illustrated in Figure 1. This optimizer enabled the model to sustain longer training sessions, especially at higher learning rates, where its averaging mechanism effectively mitigated the "jumps" characteristic of SGD. Table 1 further highlights this effect, demonstrating NTAvSGD's ability to stabilize training under aggressive learning rate schedules.

4. References

[1] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," in *Proceedings of the 26th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1993, pp. 313–330.

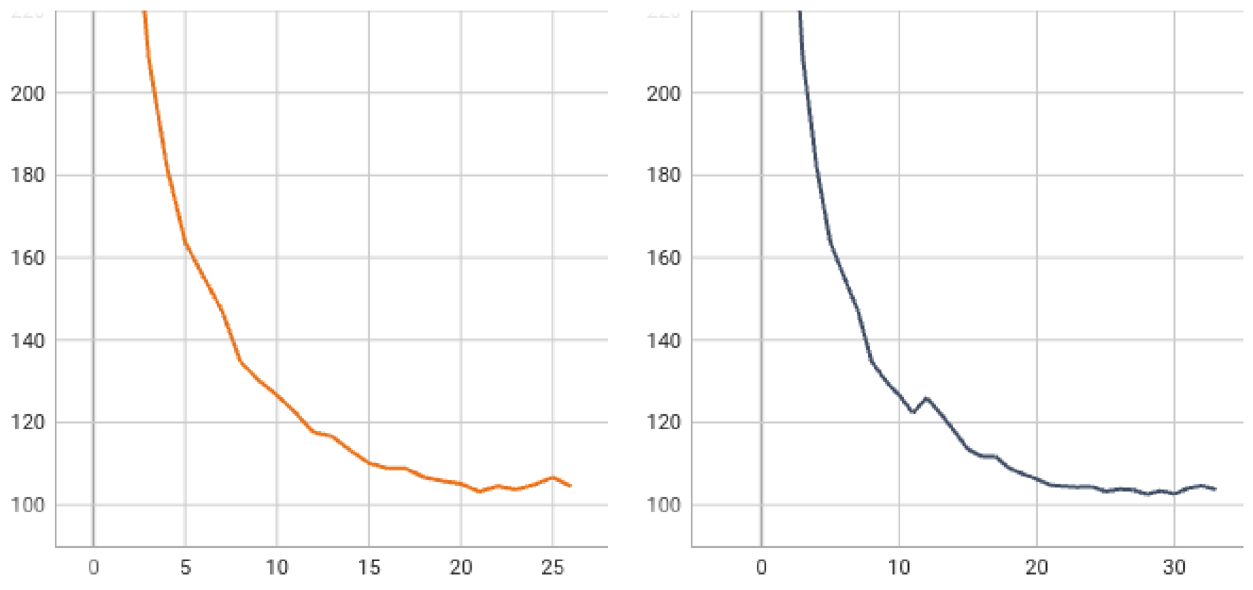
[2] O. Press and L. Wolf, "Using the output embedding to improve language models," 2017. [Online]. Available: https://arxiv.org/abs/1608.05859

[3] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," 2016. [Online]. Available: https://arxiv.org/abs/1512.05287

[4] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," 2017. [Online]. Available: https://arxiv.org/abs/1708.02182

Part	Experiment	LR	Epochs	Dev PPL	Test PPL
1.0	Baseline05	0.5	47	173.4	169.1
1.0	Baseline10	1.0	37	**165.9**	**158.9**
1.0	Baseline20	2.0	14	180.1	179.4
1.1	LSTM05	0.5	99	160.5	155.3
1.1	LSTM10	1.0	62	155.1	151.6
1.1	LSTM20	2.0	35	153.3	151.9
1.1	LSTM30	3.0	27	152.2	**147.7**
1.1	LSTM40	4.0	21	**150.6**	148.7
1.2	LSTM30-DropEmb	3.0	50	148.3	145.6
1.2	LSTM30-DropOut	3.0	52	**127.8**	**125.4**
1.2	LSTM30-Drop	3.0	72	147.2	142.7
1.3	LSTM-Drop-AdamW	0.001	31	**131.7**	**121.3**
1.3	LSTM-Drop-AdamW	0.003	15	132.5	124.1
2.0	WT20	2.0	69	130.6	128.6
2.0	WT30	3.0	44	130.4	**127.5**
2.0	WT40	4.0	43	**128.4**	127.8
2.1	30VarDropEmb	3.0	57	124.1	121.7
2.1	20VarDropOut	2.0	61	132.9	131.1
2.1	30VarDropOut	3.0	38	139.0	135.4
2.1	40VarDropOut	4.0	61	111.5	109.9
2.1	50VarDropOut	5.0	26	**103.2**	**102.3**
2.2	50DropOut-NTAvSGD	5.0	33	**102.5**	**100.4**

Table 1: *Performance of the models for each configuration.* ***Bold*** *values represent the best model for each section, underline the best overall.*



(a) *PPL/Dev on the baseline.*

(b) *PPL/Dev using NTAvSGD.*

Figure 1: *PPL/dev curve comparison between baseline and averaged model. The model is averaged after the* 11^{th} *epoch, it's evident how averaging the weights stabilizes the model leading to a slower learning and a smoother curve.*