NLU course projects - Assignment 1 - LM

Ettore Saggiorato (247178)

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

1. Introduction

This report outlines the implementation and evaluation of various techniques designed to enhance model performance in a next-word prediction task. The work is conducted as part of the first assignment for the Natural Language Understanding course at the University of Trento. The primary objective is to incrementally modify a baseline Recurrent Neural Network (RNN) model and analyze the impact of these modifications on performance. Notably, changes may result in either improvements or degradations in the model's effectiveness.

Model performance is assessed using Perplexity (PPL), a standard evaluation metric in language modeling tasks. 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: Long Short-Term Memory (LSTM) networks are implemented as a replacement for the standard RNN architecture. LSTMs are particularly effective for handling long sequential data because they address the vanishing gradient problem through their gated mechanisms [2]. This modification is expected to improve both performance and convergence speed.

2.2. Regularization

The following regularization techniques are applied to enhance model generalization and robustness:

- **Dropout:** Dropout layers are introduced at two strategic points in the network—after the embedding layer and after the final linear 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 [3].
- **Variational Dropout:** Variational Dropout [4] applies a consistent dropout mask across the entire sequence within a batch. 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:** AdamW is a more advanced optimizer than SGD. It leverages adaptive learning rates and weight decay to achieve faster convergence. Consequently, it enables the model to reach optimal solutions more efficiently.
- **NT-AvSGD:** NT-AvSGD [5] is designed to stabilize SGD, which often exhibits erratic behavior during training. By averaging model weights when validation performance plateaus, NT-AvSGD reduces instability. The optimizer incorporates mechanisms to determine when averaging should begin and how to compute the averages, leading to improved resilience against overfitting, especially at higher learning rates.

3. Results

3.1. Architecture

Replacing the RNN with an LSTM architecture resulted in significant performance gains, as shown in Table 1. The LSTM not only achieved lower perplexity (PPL) compared to the baseline RNN but also exhibited faster convergence during training.

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 significant overfitting, leading to improved results. However, this came at the cost of increased training time.

3.3. Optimizers

- **AdamW:** AdamW exhibited superior convergence speed, achieving comparable or better results than SGD with fewer training epochs.
- **NT-AvSGD:** NT-AvSGD 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 NT-AvSGD'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] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

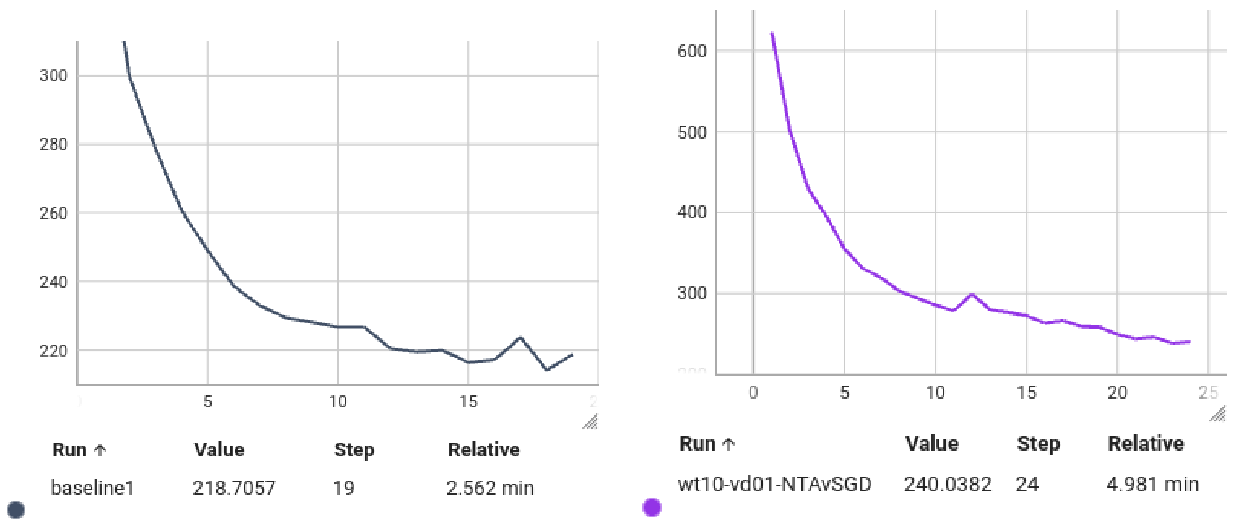
[3] O. Press and L. Wolf, "Using the output embedding to improve language models," 2017. [Online]. Available: https://arxiv.org/abs/1608.05859

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

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

Part	Model Name	LR	Epochs	Dev PPL	Test PPL
1.0	baseline05	0.5	30	*213.3*	199.2
1.0	baseline10	1.0	19	218.7	*198.6*
1.0	baseline20	2.0	10	242.0	225.0
1.0	baseline30	3.0	11	304.6	269.4
1.1	lstm05	0.5	24	222.9	207.3
1.1	lstm10	1.0	19	210.0	194.0
1.1	*lstm20*	2.0	18	*197.5*	*183.6*
1.2	lstm_drop_in_20	2.0	22	208.6	191.5
1.2	lstm_drop_in_50	2.0	17	243.6	224.5
1.2	lstm_drop_out_20	2.0	21	194.7	180.8
1.2	***lstm_drop_out_50***	2.0	29	***190.0***	***178.2***
1.2	lstm_drop_25	2.0	19	208.9	194.7
1.2	lstm_drop_50	2.0	23	244.0	224.7
1.3	lstm_adam_001	0.001	12	211.9	182.7
1.3	lstm_adam_003	0.003	11	208.1	182.4
1.4	*lstm_adam_001_drop_50*	0.001	13	*204.8*	*180.1*
1.4	lstm_adam_003_drop_50	0.003	9	229.2	193.0
2.1	wt05	0.5	28	259.3	243.7
2.1	wt1	1.0	26	214.3	202.4
2.1	*wt2*	2.0	19	*207.0*	*191.5*
2.2	wt05-vd01	0.5	35	255.0	237.3
2.2	wt05-vd02	0.5	36	265.4	241.2
2.2	wt05-vd03	0.5	34	269.8	256.1
2.2	wt10-vd01	1.0	34	213.9	200.8
2.2	wt10-vd02	1.0	32	225.3	210.1
2.2	wt10-vd03	1.0	28	235.4	223.4
2.2	*wt20-vd01*	2.0	26	*203.4*	*187.5*
2.2	wt20-vd02	2.0	23	216.4	204.6
2.2	wt20-vd03	2.0	35	214.8	203.4
2.3	wt05-vd01-NTAvSGD	0.5	36	265.2	251.8
2.3	wt10-vd01-NTAvSGD	1.0	24	240.0	227.0
2.3	wt20-vd01-NTAvSGD	2.0	21	209.9	200.2
2.3	***wt30-vd01-NTAvSGD***	3.0	22	***201.9***	***186.6***

Table 1: *Performance of the models for each configuration. Italic values represent the best model for each section,* **bold** *the best for each part of the assignment.*



Run ↑	Value	Step	Relative
● baseline1	218.7057	19	2.562 min

Run ↑	Value	Step	Relative
● wt10-vd01-NTAvSGD	240.0382	24	4.981 min

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

(b) *PPL/Dev using NTAvSGD.*

Figure 1: *PPL/dev curve comparison between baseline and averaged model.*