Language Modelling - Lab 4

Marco Prosperi (257857)

University of Trento

marco.prosperi@studenti.unitn.it

1. Introduction

The first part of the project aimed to improve the baseline RNN performance by incrementally adding features and dynamically adjusting hyperparameters (learning rate, hidden units, embedding size). The model was trained on the Penn Treebank dataset, using perplexity as the main evaluation metric. In the second part, advanced techniques [1] were incorporated to further enhance the language model, including Weight Tying to reduce parameters, Variational Dropout (excluding DropConnect) for regularization, and Averaged Stochastic Gradient Descent (AvSGD) for better optimization. These improvements aimed to enhance generalization and convergence while maintaining low perplexity (less than 250).

2. Implementation details

The first part of the project involved implementing a basic RNN model using PyTorch, based on the code provided by the teaching assistant in the Lab Repository [2]. I started by integrating the functions, model, and dataset from the Jupyter notebook into my own repository. Initially, the RNN model was trained on the Penn Treebank dataset, yielding a high perplexity that did not meet the target of 250. To address this, I experimented with various hyperparameters, including the learning rate, hidden units, and embedding size, eventually achieving an acceptable perplexity. Subsequently, I developed an LSTM model, which significantly lowered the perplexity to 137.31. To further enhance performance, I incorporated two dropout layers, resulting in better model accuracy. Additionally, the adoption of AdamW, a weight decay optimizer, further reduced the perplexity to 105.46. Given the consistent performance improvements, I retained these regularization techniques for the second part of the project.

In the second phase, I focused on implementing advanced techniques to further optimize the language model. The first enhancement was Weight Tying, which decreased the model's parameter count and improved generalization. I also introduced Variational Dropout [3], a method specifically designed for recurrent neural networks that applies a fixed dropout mask across the entire input sequence, enhancing temporal consistency and training stability.

Finally, I employed the Non-monotonically triggered AvSGD [1], which switches to AvSGD when SGD converges to a steady-state distribution. The technique can be activated when model performance plateaus, as recommended by the authors: AvSGD is initiated when the validation metric does not improve over multiple cycles. A practical implementation involves using a patience mechanism similar to early stopping; once patience is exhausted, AvSGD is triggered, and patience is reset to enable early termination. Alternatively, the switch can occur when validation loss exceeds the minimum observed within a non-monotonic window, a strategy inspired by the official Salesforce Research repository [4].

3. Results

The results of my experiments are summarized in Table 1.

In the first phase of the project, I started with a basic RNN model and experimented with different hyperparameters. After some tuning, the model achieved a perplexity of 173.22. Subsequently, I implemented several techniques that contributed to improved performance in each step. The validation perplexity is shown in Figure 2, which illustrates the results of the LSTM model with dropout layers and AdamW optimizer.

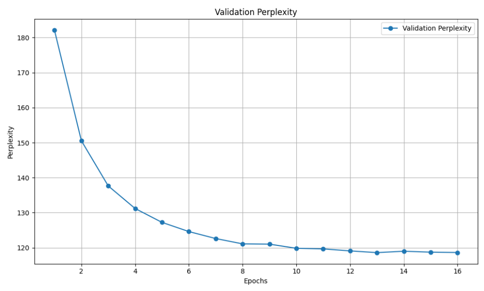

Figure 1: *Validation perplexity of the LSTM model with dropout layers and AdamW.*

In the second phase, I focused on implementing advanced optimization techniques, building upon the best model from the previous step. This improved model achieved a perplexity of 90.56, with a learning rate of 2, and both hidden and embedding units set to 400. One important observation was that the performance of the model improved when using a learning rate scheduler that decreased the learning rate over time, rather than relying on a fixed learning rate. Specifically, I used a StepLR scheduler with a step size of 5 and a gamma of 0.75.

Additionally, I observed that when AvSGD was triggered, the training loss increased temporarily. This behavior stems from the shift in optimization dynamics: when switching

from standard SGD to AvSGD, the model begins averaging past parameter updates, which can disrupt the immediate optimization trajectory. While this averaging improves long-term stability and generalization, it may initially raise the training loss as the model adapts to the new update regime.

Furthermore, the learning rate scheduler (StepLR) compounds this effect—when the learning rate is reduced, the model's updates become more conservative, requiring additional steps to re-stabilize convergence. The temporary loss increase reflects this adjustment period, where the model transitions toward a flatter, and often more generalizable, minimum in the loss landscape.

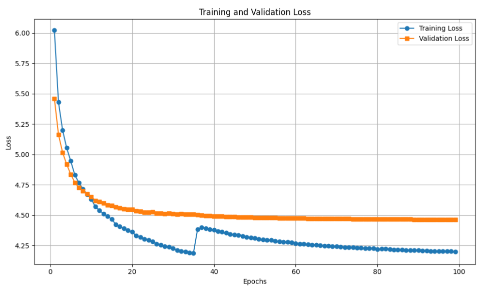

Figure 2: *Trigger point of AvSGD*

Model	**PPL**	**LR**	**Hidden**	**Emb**
Part A				
RNN	173.22	0.1	100	100
LSTM	137.31	2	300	300
LSTM + Dropout Layers	123.14	2	300	300
LSTM + Dropout Layers + AdamW	105.46	0.001	250	300
Part B				
LSTM + Weight Tying + Var Dropout + AvSDG	90.56	2	400	400

Table 1: *Perplexity and hyperparameters of the best models.*

4. References

[1] S. Merity, N. S. Keskar, and R. Socher, “Regularizing and Optimizing LSTM Language Models,” *arXiv preprint arXiv:1708.02182*, 2017.

[2] UniTrento, “Natural language understanding labs,” 2025. [Online]. Available: https://github.com/BrownFortress/NLU-2025-Labs/tree/main

[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] SalesforceResearch, “Salesforce lstm reasearch repository,” 2018. [Online]. Available: https://github.com/salesforce/awd-lstm-lm