NLU course projects: Lab. 4 (LM)

Alessandro Lorenzi (247177)

University of Trento

alessandro.lorenzi-1@studenti.unitn.it

1. Introduction

This report contains the work done to solve the **LM task**, in particular to implement and improve the performance of different architectures for the **next-word prediction** task. The comparison is done, in terms of architecture, between the RNN and LSTM and, for optimizers, between SGD, AdamW and Monotonically Triggered Average SGD. Regarding the regularization instead, dropout, weight tying and variational dropout are part of the experiments. A lot of effort was also done in the hyperparameter optimization, specially for the learning rate.

2. Implementation details

The architectures and all the methods used are then briefly described. All the methods are added incrementally, step by step.

2.1. Models

The two architectures used are the Recurrent Neural Network (RNN) and Long short-term memory (LSTM), that use gates to regulate better the flow of information through the unit. Both are suitable neural model to solve the considered task and are already implemented in PyTorch.

2.2. Regularization

Some regularization techniques are investigated, in particular:

- Dropout: technique for neural networks that drops a unit (along with connections) at training time with a specified probability p. In practice, a new dropout mask is sampled independently for each forward pass through the module. This means that the dropout mask applied to each time step in the sequence can be different.
- Weight tying [1]: it reduces the overall parameter count by sharing weights between the embedding and softmax layers. This approach avoid the need for the model to learn a direct one-to-one mapping between input and output, leading to significant improvements in the standard LSTM language model.
- Variational Dropout: method that repeat the same dropout mask at each time step for both inputs, outputs, and recurrent layers (drop the same network units at each time step). This is in contrast to ordinary Dropout where different dropout masks are sampled at each time step for the inputs and outputs alone. This ensures consistency in dropout patterns across time, which is essential for effective regularization in recurrent neural networks.

2.3. Optimization

The optimizers used for the experiments are the SGD and the AdamW, in their standard versions, implemented in PyTorch. Furthermore, the Non-monotonically Triggered AvSGD (NT-AvSGD) is implemented from scratch, inheriting from SGD, and tested. It is a modified version of standard Averaged SGD (AvSGD) that uses non-monotonic triggers to decide when to average model parameters. The implementation follows [2].

3. Results

The metric evaluated is the perplexity (PPL), it quantifies the model's "surprise" when encountering new data — lower surprise indicates better prediction accuracy.
Table 1 and Table 2 show the results of all the experiments conducted in the first and second parts. Figure 1 and Figure 2 show the PPL Dev curves, reporting first the best model of each experiment, and then with more focus on the relationship between the architecture and the learning rate set.

3.1. Models

The experiments conducted demonstrate that adding methods that handle the vanishing gradient problem to the architectures, improve the results. The most notable scores were indeed observed with the LSTM NTAvSGD model, that combine adaptive lr, weight tying and variational dropout.

3.2. Regularization

Incorporating regularization techniques, specifically variational dropout, has proven beneficial for improving model generalization. The results highlight that variational dropout helps mitigate overfitting, leading to better performance on both training and validation sets.

3.3. Optimization

The choice of optimization algorithm played a pivotal role in the models' performance. The adaptive NTAvSGD optimizer outperformed the standard SGD in conjunction with weight tying, particularly evident at higher learning rates.

3.4. Learning Rate

The learning rate is a critical hyperparameter that significantly influences model convergence and overall performance. In the experiments, various learning rates were tested across different models, revealing that higher learning rates initially produced lower perplexity scores. It is evident that excessively high learning rates can lead to instability and suboptimal performance. Conversely, lower learning rates may improve stability but could slow down convergence.

4. References

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

[2] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," 2017.

Architecture	Learning Rate (lr)	PPL / Test	PPL / Dev	Epochs
RNN SGD	0.5	184.0	191.50232	42
	1.0	**161.454**	**170.17566**	46
	1.5	164.305	175.96648	29
	2.0	169.83888	179.50621	27
LSTM SGD	0.5	167.578	174.76499	93
	1.0	153.831	160.58694	69
	1.5	149.775	159.62784	55
	2.0	**149.21**	**156.67388**	38
LSTM SGD drop (p=0.3)	1.0	144.696	150.3115	136
	1.5	139.207	144.58829	109
	2.0	**130.739**	**135.68674**	123
LSTM AdamW drop (p=0.3)	0.0005	**112.622**	**122.3638**	89
	0.001	113.338	122.58145	53
	0.002	114.843	123.00774	28

Table 1: *Part 1 Results. The terms in **bold** indicate the best performance obtained for the corresponding configuration.*

Architecture	Learning Rate (lr)	PPL / Test	PPL / Dev	Epochs
LSTM SGD weight tying	1.0	128.668	133.43789	124
	1.5	129.624	133.22032	83
	2.0	**125.727**	**130.77683**	73
LSTM SGD weight tying + variational dropout	1.0	120.481	124.89424	158
	2.0	104.566	107.06002	162
	3.0	**105.898**	**110.11653**	96
LSTM NTAvSGD weight tying + variational dropout	1.0	114.646	117.85777	199
	2.0	102.082	109.92505	199
	3.0	**<u>101.061</u>**	**<u>103.72644</u>**	199
	4.0	103.104	105.61212	199

Table 2: *Part 2 Results. The terms in **bold** indicate the best performance obtained for the corresponding configuration, while the <u>underlined</u> term represents the best overall results (considering both parts).*

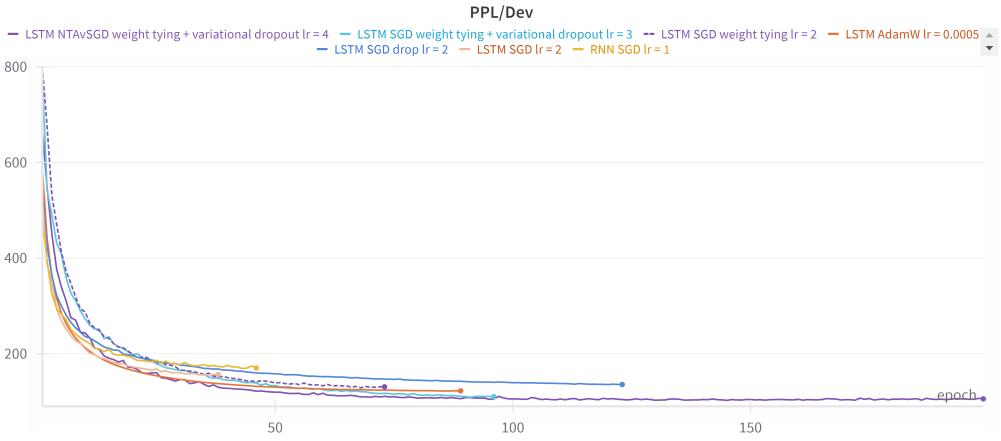
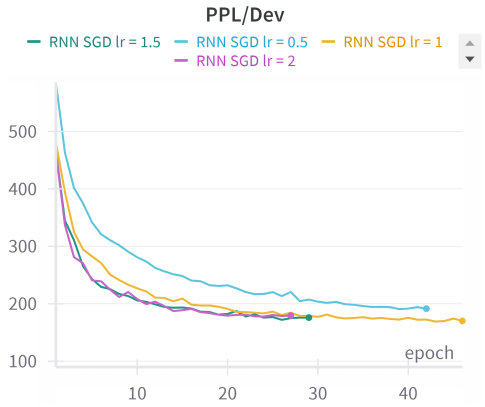
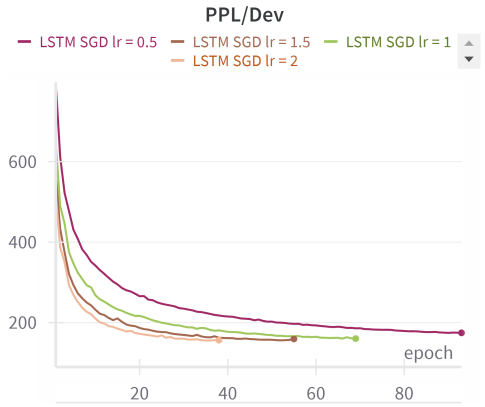


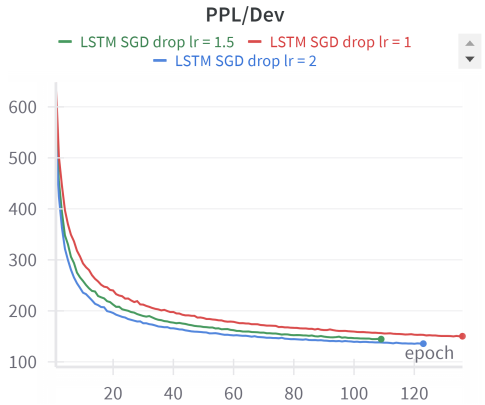
Figure 1: *PPL Dev results (only the best model of both part 1 and 2 are shown)*



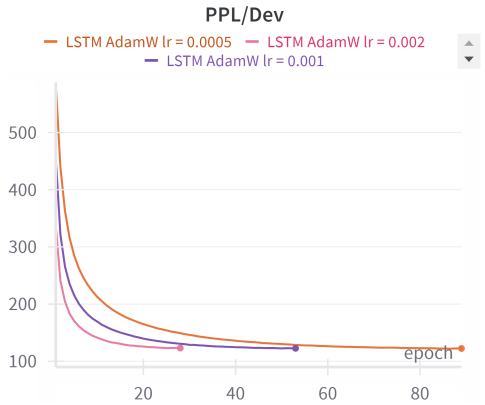
(a) *RNN SGD*



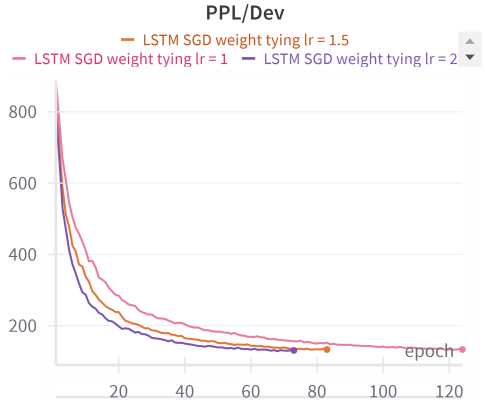
(b) *LSTM SGD*



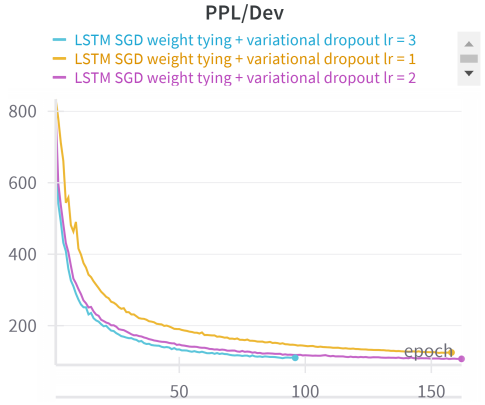
(c) *LSTM SGD dropout*



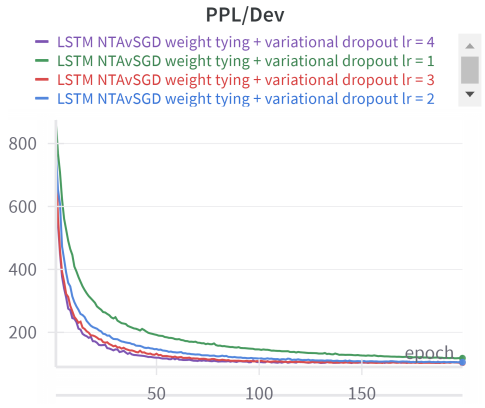
(d) *LSTM AdamW dropout*



(e) *LSTM SGD weight tying*



(f) *LSTM SGD weight tying + variational dropout*



(g) *LSTM NTAvSGD weight tying + variational dropout*

Figure 2: *Performance of all the models with the differents lr*