Lab 4: Improving a Recurrent Language Model

Filippo Adami (mat. 256149)

University of Trento

name.surname@studenti.unitn.it

1. Introduction

In this report, I document the process of incrementally improving a baseline RNN Language Model (LM). I first performed extensive hyperparameter tuning on the baseline RNN to establish a strong starting point. I then incrementally replaced the RNN with an LSTM, added standard dropout layers, and switched the optimizer from SGD to AdamW. Later, starting from a base LSTM model I applied a set of advanced regularization techniques: Weight Tying, Variational Dropout, and Non-monotonically Triggered AvSGD. My goal at each step was to minimize the perplexity (PPL) on the validation set.

2. Implementation details

I began with the provided baseline RNN implementation from Lab 4, which I edited and enriched step by step to follow the guidelines of the project. I evaluated each model evaluated after every epoch, and I used early stopping to prevent infinite training (as provided in the lab code). For each main step, the model with the best performance was saved, while evaluation metrics were saved for all the tried models.
I implemented a modular approach to manage the different settings without losing clarity in the code, and to allow easy refinement and debugging.
I used loops to streamline hyperparameter tuning and to optimize computational costs. Once a loop was finished, I reviewed the results to decide whether additional manual intervention was necessary to get the best values for some hyperparameters.
I tried each hyperparameter only within a certain range of values to keep a balance between expected performance and expected time/memory/computation costs. I chose these ranges based on personal knowledge gained both in class and via online resources.
My very first step was to tune the hyperparameters of the RNN (mainly learning rate and batch size) to establish a strong baseline and moved from an initial PPL of 5768.69 to 151.

$\mathbf{H}_{dimension}$	$\mathbf{E}_{size}$	$\mathbf{B}_{size}$	LR	PPL
200	300	64	0.0001	5769
200	300	64	1.0	158
200	300	64	0.5	155
200	300	64	0.1	180
200	300	16	1.0	158
200	300	16	0.5	168
200	300	32	1.0	157
200	300	32	0.5	155
200	400	32	0.5	154
400	400	32	0.5	**151**

Table 1: *Summary of RNN Hyperparameter Tuning Results*

3. Results

3.1. Part 1.A

- **LSTM:** Replacing the 'RNN' with an 'LSTM' required re-tuning the learning rate, as the LSTM's gating mechanisms allows for a much higher stable SGD learning rate. Furthermore, the extra gradient noise from smaller batches acts as regularization, improving stability.

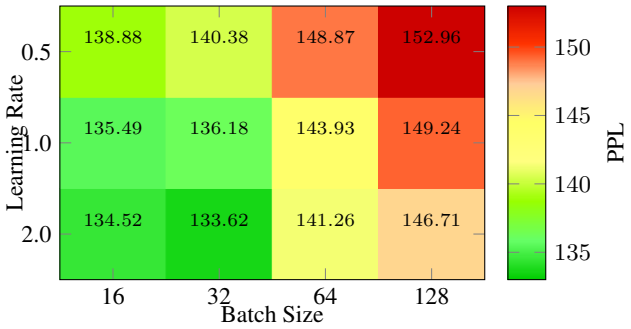

Figure 1: *PPL Heatmap of tried LSTM models*

- **Dropout:** I added two Dropout layers: one after the embedding layer (smaller dropout rate to avoid 'corruption' of input sequences) and one before the final linear layer (higher dropout rate to prevent the classifier head from memorizing training patterns).

$\mathbf{E}_{dropout}$	$\mathbf{OUT}_{dropout}$	PPL
0.1	0.3	115.07
0.1	0.5	109.99
0.1	0.7	107.02
0.2	0.3	115.44
0.2	0.5	109.97
0.2	0.7	**106.77**

Table 2: *Perplexity for dropout configurations.*

- **AdamW:** I replaced the SGD optimizer with 'AdamW'. This required a significant re-tuning of the learning rate, as my rates that perform well with SDG are too high and cause divergence with AdamW.
Furthermore, AdamW allowed for larger embedding size and hidden dimension. I then fixed dropout rates accordingly.

LR	$\mathbf{E}_{dropout}$	$\mathbf{OUT}_{dropout}$	$\mathbf{H}_{dim}$	$\mathbf{E}_{size}$	PPL
2e-2	0.2	0.7	400	400	348.88
3e-3	0.2	0.7	400	400	119.04
4e-4	0.2	0.7	400	400	112.41
4e-4	0.2	0.7	800	600	106.37
4e-4	0.2	0.7	800	700	108.24
4e-4	0.3	0.65	800	600	**101.98**
4e-4	0.3	0.8	800	600	105.20

Table 3: *Perplexity for AdamW under varying learning rates, model sizes, and dropout rates.*

3.2. Part 1.B

Starting from the base LSTM model, I explored the effects of some advanced techniques.

- **Weight Tying:** I implemented 'Weight Tying' by setting the dimension of the final linear decoder to be the same as the size of embedding.

Dimension	PPL
400	116.62
300	**115.43**
200	118.59

Table 4: *LSTM + Weight Tying perplexity results*

- **Variational Dropout:** I then implemented Variational Dropout for the LSTM on the same layers as in point 3.1. This required a re-tuning of both the dropout rates and the dimension parameter.

Dimension	$\mathbf{E}_{dropout}$	$\mathbf{OUT}_{dropout}$	PPL
300	0.1	0.70	105.15
300	0.3	0.65	97.46
300	0.1	0.30	101.66
500	0.2	0.50	95.93
700	0.3	0.70	**91.44**
800	0.3	0.70	95.02

Table 5: *LSTM + Weight Tying and Variational Dropout perplexity results*

- **NT-AvSGD:** Finally, I switched optimization strategy to Non-monotonically Triggered Averaged SGD. This allowed for a higher learning rate.

Dimension	$\mathbf{E}_{dropout}$	$\mathbf{OUT}_{dropout}$	LR	PPL
700	0.7	0.7	3	**87.55**
700	0.5	0.7	3	91.37
700	0.2	0.6	3	95.58
700	0.2	0.6	1	98.65
500	0.2	0.6	3	95.24
500	0.2	0.6	2	96.43

Table 6: *LSTM + Weight Tying, Variational Dropout, and NT-AvSDG results*