Regularizing and Optimizing RNN Language Models

Soyoung Yoon
20160413

Jaeyoung Hwang
20150824

Dongmin Seo
20150390

Abstract

Regularizing and Optimizing LSTM Language Models [3] introduce regularizing and optimizing techniques that do not need modifying original LSTM based model. But in order to use this Language Model in real-life, model has to meet several space and time limitations. Therefore, We propose to make the model further lightweight and faster to train without much giveaway on perplexity. For example, by changing the model from LSTM to GRU, or from unidirectional LSTM to bidirectional LSTM. As a result, we successfully replicated the original paper's model which shows almost original perplexity. Then, we applied these techniques on GRU and biLSTM. We concluded that the GRU model is faster to train, but LSTM models show better perplexity results than the GRU model. Also biLSTM models actually take longer time to train, and they give extraordinary results. From this experiment we learned that we have to train and evaluate bidirectional LSTM models in a different way. We opensourced our code in github. [1]

1 Introduction

Language modeling is useful for pre-training decoders in Seq2Seq architectures, and custom architectures often proposed. *Regularizing and Optimizing LSTM Language Models* [3] introduce some techniques that do not need modifying original RNN model. This paper applies several regularizing and optimizing techniques, such as weight drop, weight tying, variational dropout, random BPTT length, AR and TAR, for increasing the model performance. Furthermore, it propose new optimization techniques(NT-AvSGD). LSTM model with those techniques attained State-of-The-Art performance for many tasks and become popular baseline model for Language Model papers.

[1] https://github.com/soyoung97/awd-lstm-gru

Nowadays, mobile is an important device for users. Therefore, many companies try to use language models on mobile devices. But ML models for real production has space limitation. Therefore, for making it lightweight and faster to train, we provide a way to maintain the performance of these neural architectures while substantially reducing the parameter cost. The paper achieves that goal by introducing regularizing and optimization techniques without additional learning parameters. GRU is also a RNN model that has fewer parameters than LSTM model although they show similar performance. Therefore, we thought GRU model will be more suitable for this purpose than LSTM model.

In this project, we experiment with those regularizing and optimizing techniques on GRU and bidirectional LSTM model.

2 Background

These are the regularizing and optimizing techniques that were used in the original paper. In addition, we introduce the details of RNN models which we changed from LSTM model.

2.1 Weight Drop

We used DropConnect [4] on the recurrent hidden to hidden weight matrices which does not require any modifications to an RNNs formulation. As the dropout operation is applied once to the weight matrices, before the forward and backward pass.

2.2 Embedding and Variational dropout

Embedding Dropout reduces total parameter size. This is used on the embedding matrix at a word level. Variational Dropout overcomes the problem of disrupting RNNs ability to retain long term dependencies by applying same dropout masks over multiple time steps.

2.3 NT-AvSGD

SGD is among the most popular methods for training deep learning models.

$$\min_w \frac{1}{N} \sum_{i=1}^{N} f_i(w)$$

Averaged SGD (AvSGD) has been analyzed in depth theoretically and many surprising results have been shown. Instead of returning the last iterate as the solution, returns $\frac{1}{K-T+1} \sum_{i=T}^{K} w_i$, where K is the total number of iterations and T ¡ K is a user-specified averaging trigger.
NT-AvSGD has similar as AvSGD but it has well defined guideline to decide T and K.

Algorithm 1 Non-monotonically Triggered ASGD (NTASGD)

Inputs: Initial point w_0, learning rate r, logging interval L, non-monotone interval n.

Initialize $k \leftarrow 0$, $t \leftarrow 0$, $T \leftarrow 0$, $logs \leftarrow []$
while stopping criterion not met **do**
    Compute stochastic gradient $\hat{\bigtriangledown} f(w_k)$ and take SGD step (1).
    if $\mathrm{mod}(k, L) = 0$ and $T = 0$ **then** Compute validation perplexity v
        if $t > n$ and
$v > \min_{l \in \{t-n, \cdots, t\}} logs[l]$ **then**
            Set $T \leftarrow k$
        end if
        Append v to $logs$
        $t \leftarrow t + 1$
    end if
end while
return $\frac{\sum_{i=T}^{K} w_i}{k-T+1}$

2.4 Random BPTT length

When use same window size at every epochs, any element divisible by window size will never have any elements to backprop into, no matter how many times you may traverse the data set. To prevent such inefficient data usage, we randomly select the window size for the forward and backward pass.

2.5 Weight tying

Weight tying is about sharing weights between input-to-embedding layer and output-to-softmax layer. By doing this, we can use only one matrix instead of using two matrices. Therefore, it prevents learning one-to-one correspondence between input and output.

2.6 AR and TAR

L2 decay can be used on the individual unit activations and on the difference in outputs of an RNN at different time steps. These strategies labeled as Activation Regularization (AR) and Temporal Activation Regularization (TAR) respectively. AR is defined as $aL_2(m \odot h_t)$ where m is the dropout mask, $L_2(\cdot) = \|\cdot\|_2$, h_t is the output of the RNN at timestep t, and a is a scaling coefficient. Using the notation from AR, TAR is defined as $\beta L_2(h_t - h_{t+1})$ where β is a scaling coefficient.

2.7 GRU

GRU is gating mechanism in recurrent neural networks. Although GRU's performance on many tasks was found to be similar to that of LSTM, it has fewer parameters than LSTM, as it lacks an output gate.

2.8 Bidirectional LSTM

Bidirectional LSTM(biLSTM) connects two hidden layers of opposite directions to the same output. The output layer can get information from backwards and forward states simultaneously. These were introduced to increase the amount of input information available to the network.

3 Method

First, we check if we successfully replicated the original baseline model. Then, to see if these regularizing and optimizing techniques are working well on the modified models, we experimented with the following changed RNN model. The baseline of our experiment is LSTM with 3 layers applied regularization and optimization techniques. Split cross entropy is used for the loss function, and perplexity is calculated by the exponential of loss. First to see if we implemented the original code well, we tested on making the "LSTM without dropout" model. Then, we increased different number of layers of GRU to see which performs better. Lastly, we implemented the bidirectional LSTM model. The following modeles are listed below.

- LSTM without dropout
- GRU with more layers(5)
- GRU with same number of layers(3)
- Bidirectional LSTM with same number of node

4 Training

We use same dropout rate used in embedding matrix, hidden to hidden layers, RNN layers in the network and each embeddings were initialized randomly. Dropout rate for input embedding layers and applied layers is 0.4, removed word rate from embedding layer is 0.1, for RNN layers is 0.25. Network is trained using Penn TreeBank and WikiText-2 on NVIDIA GTX 1080 ti GPU.

We train our model through two separated stages, initial training and fine-tuning. We use weight tying, weight dropout for hidden to hidden matrix and dropout for input embedding layer and RNN layers first. After proper learning is done, we apply AR and TAR, use random back propagation through time length and use non-monotonic trigger for optimization.

5 Result

Type of Model	Pen TreeBank	Wiki Text 2
LSTM	24 M, 48 s	33 M ,91 s
GRU	19 M, 41 s	28 M, 76 s
GRU-5 layers	35 M, 76 s	-
biLSTM	18 M, 58 s	28 M, 96 s

Table 1: Total number of parameters and training time per one epoch according to dataset and model

As we planned to do, the parameter size of each model were different. Trivially, GRU with 3 layer has less parameters compared to the baseline LSTM. Bidirectional LSTM has same number of node as LSTM, but has less parameters, because connection layers between decoder and encoder of biLSTM doesn't need much parameters than hidden to hidden layers of LSTM, but more execution time is needed for each epoch, so learning time is longer than baseline LSTM. GRU-5 layers had the biggest number of parameters, showing that increasing the number of layers doesn't always help at increasing the model performance. The figure of training and validation loss over epoch of each model is listed on the Appendix(last page) of the report. Figure 1 shows that regularization methods used actually works - meaning that we successfully replicated the baseline LSTM model. LSTM without regularization techniques decreases training loss rapidly, and the validation loss is increased, meaning that it overfits to the training data. Since there is less generalization ability, it adapts to the data set too quickly. As a result, validation loss change verifies LSTM without regularization is not well trained.

The performance of GRU were lower than LSTM over both datasets. Compared with LSTM, average time to run over one epoch training time is faster for GRU with same number of layers. During the poster session we discussed that if we fine-tune the hyperparameters, the model performance would increase. After the poster session, we changed some hyper parameters, such as dropout rate for embedding matrix and RNN layers, but there is no big difference compared to the gap of LSTM loss. To testify that the cause of the problem is because of GRU problem, we train GRU again with more layers. As the figure 3 shows, GRU with more parameters than original LSTM perform worse than LSTM. In our conclusion, because of architectural reason, GRU has short memory propagation length and fewer transferred information between layers than LSTM, therefore it is not proper to language modeling task.

As we can see in the result figures, GRU has perturbation at inital training, because few parameters of GRU get low impact of gradient vanishing, so parameters are variable than LSTM. Training loss of GRU in figure 2 increase after loss converge to some stable state, since learning rate is relatively too big to settle minima of loss function. It is need to train again with lower learning rate later.

For Bidirectional LSTMs, they showed perplexity scores of around 1. Bidirectional LSTMs showed unbelievable performance on our language modeling task. Although it is said that bidirectional models have great performance over other ones, compared to the proposed LSTM model which has single model perplexity of 60, having a perplexity of nearly 1 or less was somewhat strange. Since the target for language modeling is just input sequence offset by one, the network cheats by using the other direction of the LSTM to know what word is coming next, thus it predicts perfectly what the next word in the sequence will be.

6 Further works

To solve the problem of biLSTM evaluation, the tasks should be modified. We should use other tasks rather than next word prediction. For example, generation tasks, translation or classification, or training a language model using masked sequence by giving a task to predict the masked word(BERT) should be used. Furthermore, actually

single-model perplexity can be used, if we apply biLSTM only on encoder in encoder-decoder structure. Also, more modifications on the model can be done to enhance the performance of Language Modeling such as attention-based language models [1] and attentive language models [2]. We tried to implement those two modifications - changing the task of Language Modeling and evaluating based on the task and modifying to make attentive Language Modeling, but we had problems integrating the ideas into the original awd-lstm code. Therefore we could not finish until we see the results.

7 Conclusion

In our project, we discuss the ideas of the paper, "Regularizing and Optimizing LSTM Language Models", and propose modifications to the original model by changing the model structure. We first successfully replicated the original paper. Then, we changed the model from LSTM to GRU, and from LSTM to biLSTM, and compared the time and perplexity results. As a result, we found out that although the performance of GRUs were lower than LSTMs for Language modeling, they end up having less parameter size and took less training time. We also learned that training time of biLSTMs were longer than the baseline model and that we cannot evaluate the single-model perplexity of biLSTMs because of the structure of biLSTMs.

8 Appendix

Next page shows training loss and validation loss over epoch of each model.

References

[1] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. Coherent dialogue with attention-based language models. *CoRR*, abs/1611.06997, 2016.

[2] Giancarlo Salton, Robert Ross, and John Kelleher. Attentive language models. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 441–450, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[3] Richard Socher Stephen Merity, Nitish Shirish Keskar. Regularizing and optimizing lstm language models. *Journal of the Association for Computing Machinery*, arXiv:1708.02182, 2017.

[4] Zeiler M. Zhang S. LeCun Y Wan, L. and R. Fergus. Regularization of neural networks using dropconnect. *In Proceedings of the 30th international conference on machine learning*, 28:1058–1066, 2013.

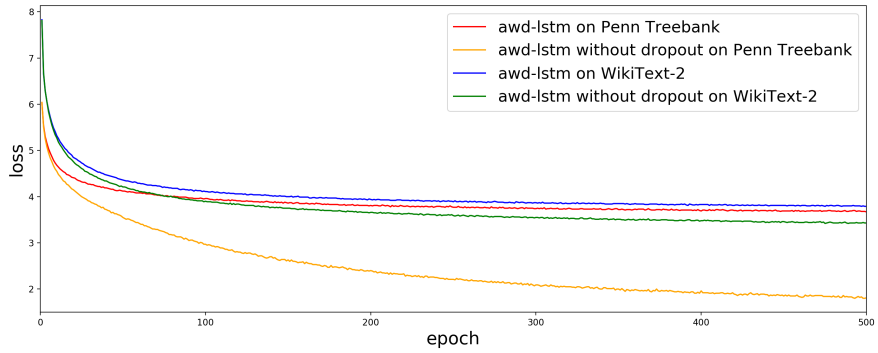

Figure 1: Train loss over PTB and WikiText-2 with baseline LSTM and baseline LSTM without dropout.

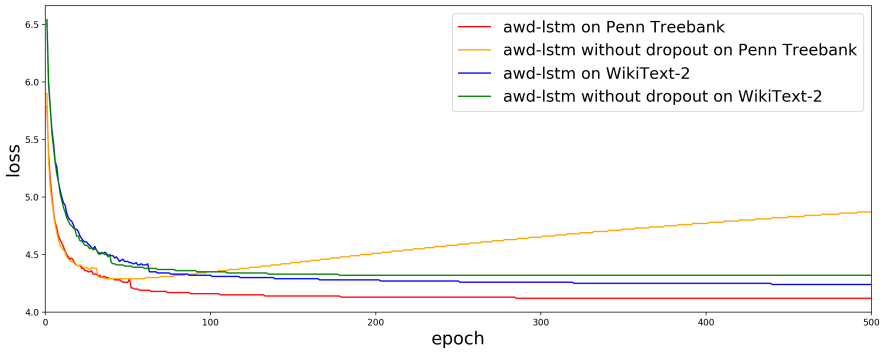

Figure 2: Validation loss over PTB and WikiText-2 with baseline LSTM and baseline LSTM without dropout.

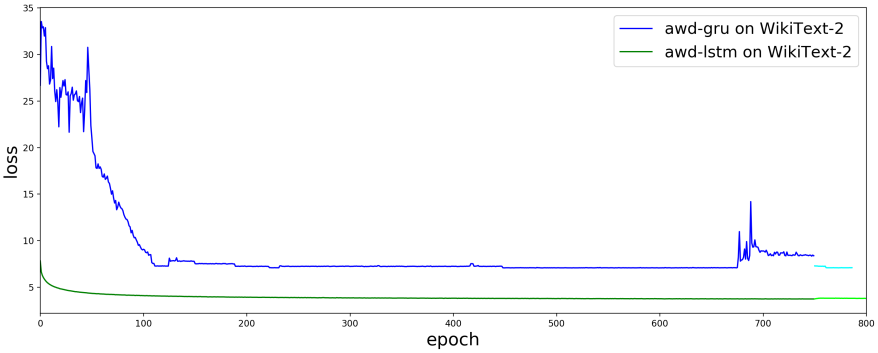

Figure 3: Train loss over WikiText-2 with GRU 3 layers and baseline LSTM.

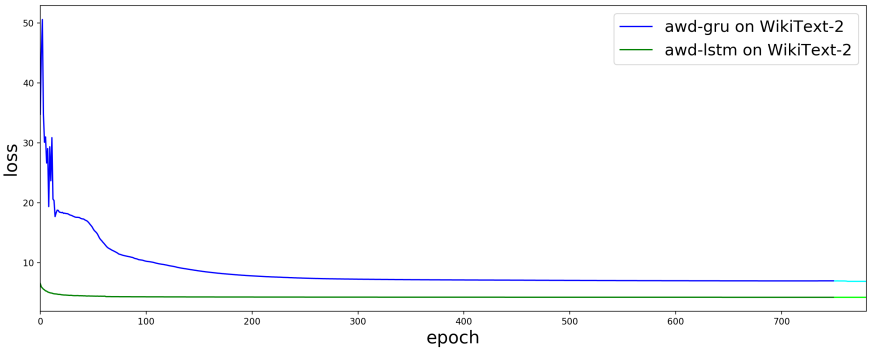

Figure 4: Validation loss over WikiText-2 with GRU 3 layers and baseline LSTM.

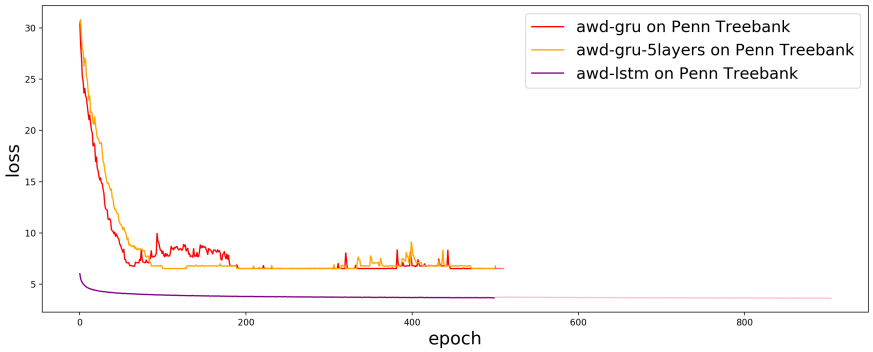

Figure 5: Train loss over Penn Treebank with GRU 3 layers and GRU 5 layers.

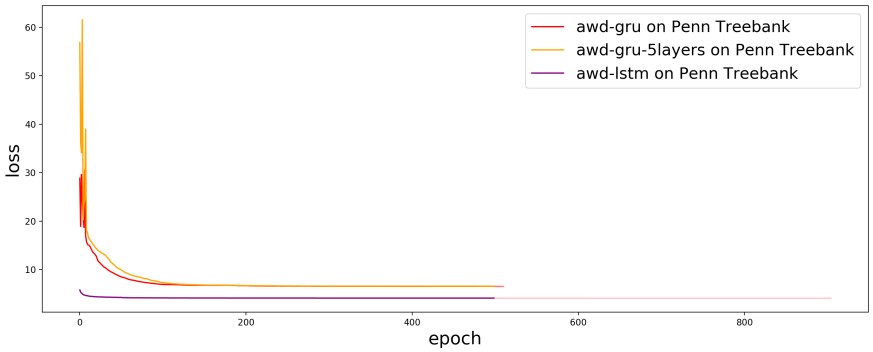

Figure 6: Validation loss over Penn Treebank with GRU 3 layers and GRU 5 layers.

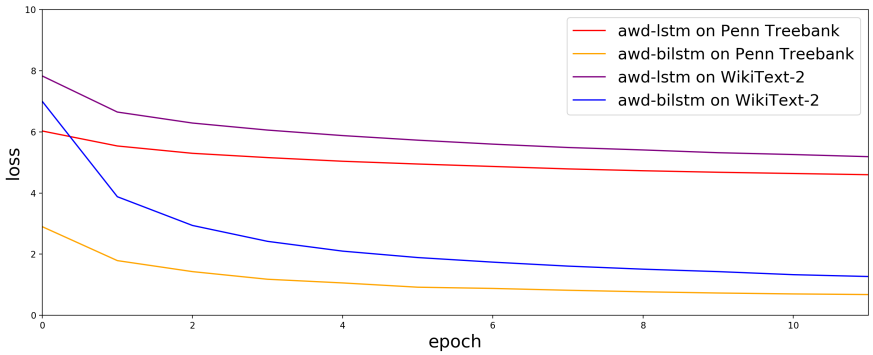

Figure 7: Train loss over Penn Treebank and WikiText-2 with biLSTM and baseline LSTM.

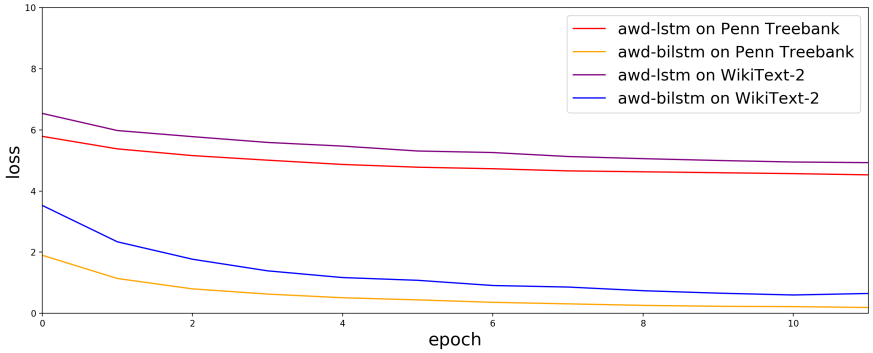

Figure 8: Train loss over Penn Treebank and WikiText-2 with biLSTM and baseline LSTM.