Yorick Chern
COMPSCI 288: NLP

Experiments Report

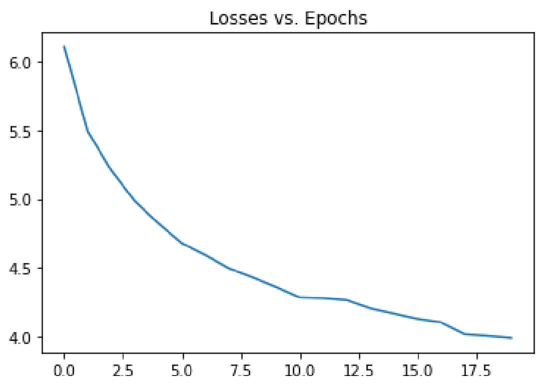
Two extensions were implemented.

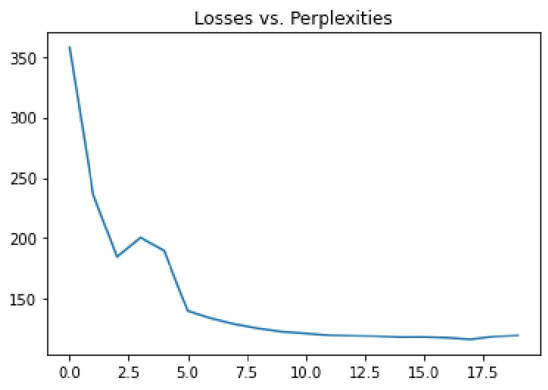
1. Increasing the hidden size of the LSTM cell from 512 → 720.
2. Implemented an L_2-regularization on the weight matrices of the LSTM cell. This code snippet shows how the L_2-regularization is implemented:

```
    l2 = 0
    for name, layer in self.network.named_parameters():
        if "lstm_layer.weight" in name:
            l2 += torch.sum(layer ** 2)

    loss = loss_fn(torch.transpose(out, 1, 2), y.T) + torch.sqrt(l2) *
1e-4
```

I wanted to implement this change because language modeling is a very complex task, and complex tasks work better with deeper and heavier networks. However, making a network bigger also implies a higher chance of overfitting, so I combat that with L_2-regularization, a technique that penalizes the weights for taking extreme values. Only the weights from LSTM are regularized because only two linear layers were used and the weights in an LSTM cell are run multiple times per one data point (depending on the sequence length) and are more prone to take on extreme values. There are also simply a lot more weights in the LSTM cell than the linear weights combined. Originally, I also added an exponential learning rate scheduler for my optimizer in hopes of better perplexity outcomes. However, the performance was worse than my baseline LSTM model, so I discarded that implementation. These two combinations were able to achieve the following learning curves:





The left visual refers to the curve with losses (y-axis) against epochs (x-axis); the right visual refers to the curve with perplexities (y-axis) against epochs (x-axis).

These techniques definitely improved the model’s performance. The baseline LSTM model reached a perplexity score of 126.28 after 20 epochs while the improved LSTM model reached a perplexity score of 116.27 after 20 epochs. Additionally, after the first epoch, the baseline LSTM model started off with a perplexity score of 432.0 while the improved LSTM model started off with a perplexity score of 351.0. This shows that the model with the extensions were able to both start off and end with better performances.

Conclusion: bigger networks, depending on the task, *almost* always boosts the performance by a bit, and the regularization definitely helped with keeping the weight matrices in a reasonable size.