Deep Learning Homework 3

Zheyu Zhang, Tsinghua University

A naive but complicated homework for DL.

Introduction

As we have learned language models in class, we're required to build models on WikiText2 [1].

The homework contains six parts.

- Build standard RNN(LSTM).
- Build standard Transformer.
- Explain how the data load from text.
- Explain why mask is necessary in Transformer and how it is implemented in my code.
- Visualize the attention of the Transformer.
- Adopt extra technique.

1 Build RNN

To build up standard RNN, I directly use *torch.nn.LSTM* [2].

The architecture is shown by Figure 1.

- Learning rate $= 2$.
- Epochs $= 800$.
- Batch size $= 32$.
- Back propagation through time $= 256$.

With the details above, the result is shown by Figure 2. The best model on validation set reaches 161.0888858709305 Perplexity on test set.

2 Build Transformer

To build up standard Transformer, I directly use *torch.nn.TransformerEncoderLayer* [3] and *torch.nn.TransformerEncoder* [4].

The architecture is shown by Figure 3.

- Learning rate $= 0.5$.
- Epochs $= 256$.
- Batch size $= 32$.
- Back propagation through time $= 256$.
- Gradient clip $= 0.005$.

Note that, in this task, I met gradient explosion. So the gradient clip is really necessary in this task while maybe not in others.

With the details above, the result is shown by Figure 4. The best model on validation set reaches 220.3037595759322 Perplexity on test set.

3 Data Processing

It is clear that, the data is simply a text.

We want to train a language model, then we simply shift one word as standard label. In detail, if we input the i-th word, then the $(i+1)$-th word is the truth to check how correct the model predict.

4 Mask Processing

Mask is necessary because the feature of data. If the Transformer can look at the word later, then the model can directly output it and learn nothing.

Figure 5 shows that how I implement it in code.

5 Attention Visualizations

I use my Transformer model trained above, and take the *attention output weights* of the first *Transformer Encoding Layer*. Figure 6 shows that the model learned some meaningful relation of words.

6 Extra Technique

I think RNN works better than Transformer on this task, so I want to use techniques to improve RNN.

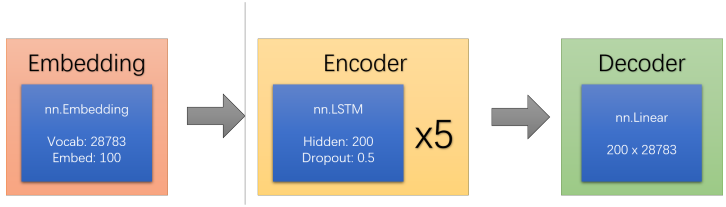

Figure 1. The architecture of my RNN.

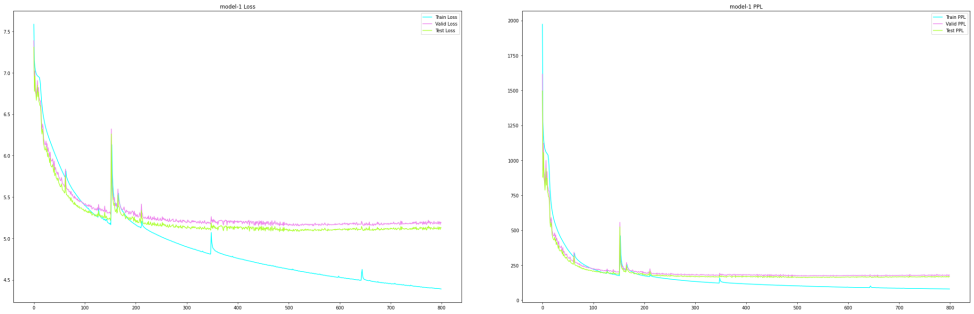

Figure 2. The result of my RNN.

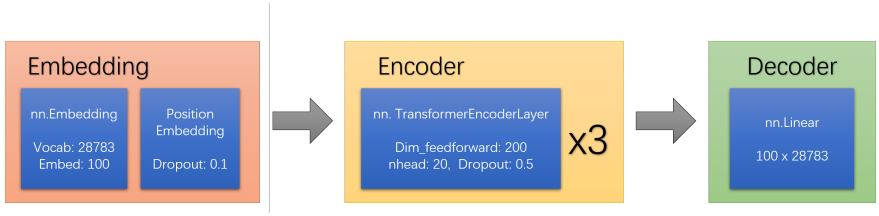

Figure 3. The architecture of my Transformer.

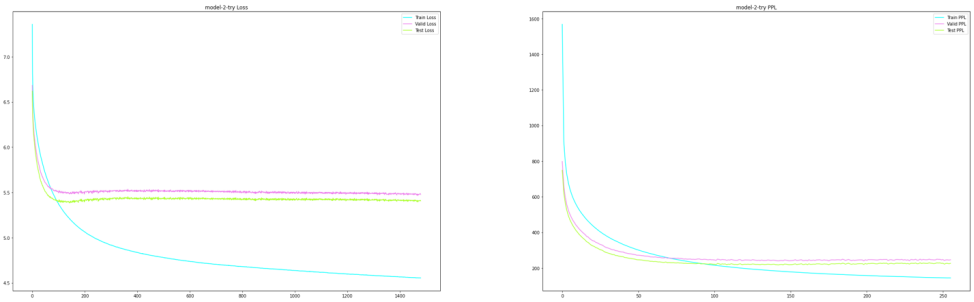

Figure 4. The result of my Transformer.

Set a function to generate mask.

```
def _generate_square_subsequent_mask(self, sz):
    mask = (torch.triu(torch.ones(sz, sz)) == 1).transpose(0, 1)
    mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
    return mask.to(device)
```

Give the mask to TransformerEncoder.

```
outputs = self.transformer_encoder(embeddings, mask=self._generate_square_subsequent_mask(embeddings.shape[0]))
```

Figure 5. Detail of implement of mask.

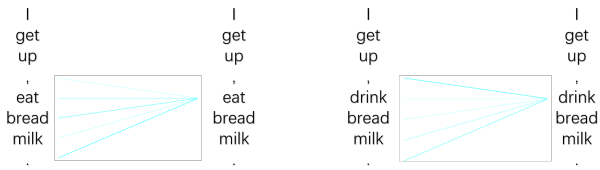

Figure 6. Transformer learned meaningful relation.

I used GLoVe [5] as the initial word embedding. There are 1028 oov(out of vocabulary) words, I remain them random value.

The training detail is exactly the same as RNN above. Figure 7 shows the result. The best model on validation set reaches 131.84482810723708 Perplexity on test set.

Acknowledgements

Thanks TencentCloud® for providing GPU.

References

[1] *Homework 3: Recurrent Neural Networks and Transformer*, Deep Learning (84100343-0), Spring 2021, Tsinghua University.

[2] *Document of torch.nn.LSTM*, Pytorch, https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

[3] *Document of torch.nn.TransformerEncoderLayer*, Pytorch, https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html.

[4] *Document of torch.nn.TransformerEncoder*, Pytorch, https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html.

[5] *Document of GLoVe*, Jeffrey Pennington, Richard Socher, Christopher D. Manning, https://nlp.stanford.edu/projects/glove/.

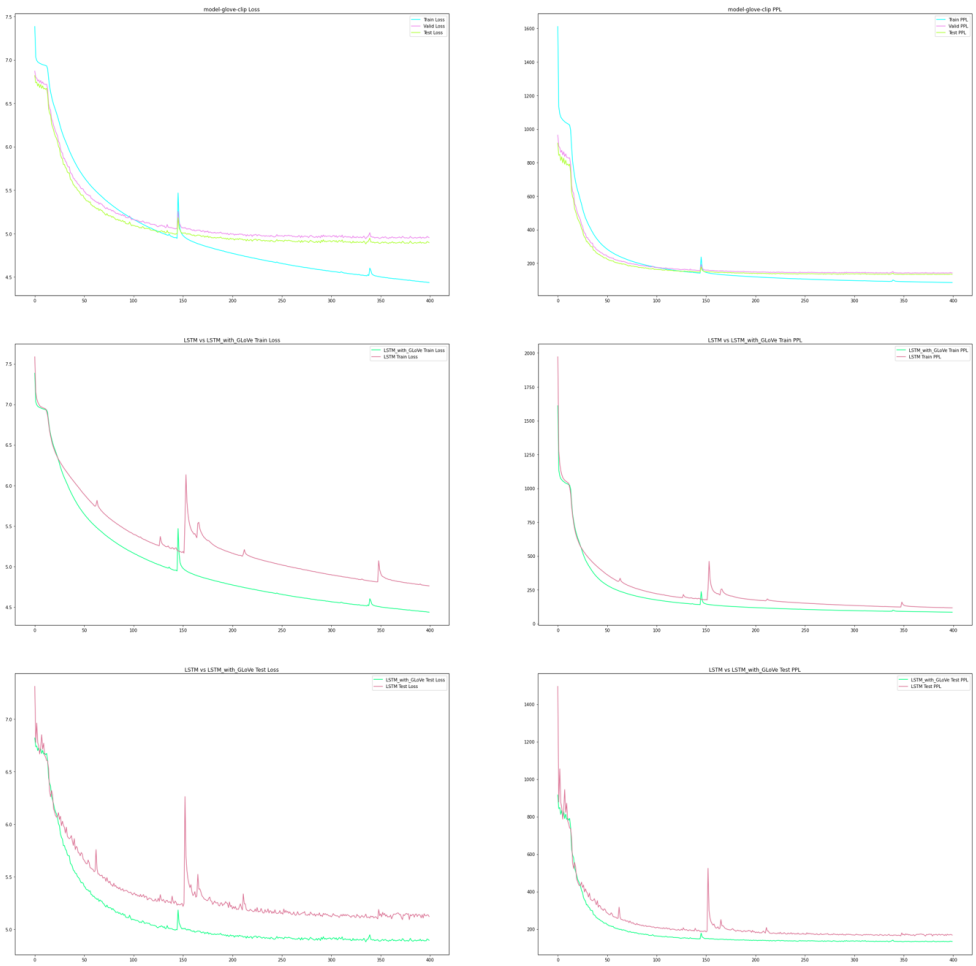

Figure 7. The result of my RNN with GLoVe.