Lab3
Machine Translation

313510164 陳緯亭 電子碩一

October 28, 2024

Contents

1 Screenshot

1.1 Parameter size

```
The parameter size of transformer is 10898.884 k
```

Fig. 1: Parameter size

1.2 Accuracy

```python
sentence = "你好，欢迎来到中国"
ground_truth = 'Hello, Welcome to China'
predicted = translate(transformer, sentence, tokenizer_cn, tokenizer_en)

print(f'{"Input:":15s}: {sentence}')
print(f'{"Prediction":15s}: {predicted}')
print(f'{"Ground truth":15s}: {ground_truth}')
print("Bleu Score (1gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 1).item())
print("Bleu Score (2gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 2).item())
print("Bleu Score (3gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 3).item())
print("Bleu Score (4gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 4).item())
```

```
Input:          : 你好，欢迎来到中国
Prediction      : You are going to welcome you.
Ground truth    : Hello, Welcome to China
Bleu Score (1gram):  0.3333333134651184
Bleu Score (2gram):  0.0
Bleu Score (3gram):  0.0
Bleu Score (4gram):  0.0
```

```python
sentence = "早上好，很高心见到你"
ground_truth = 'Good Morning, nice to meet you'
predicted = translate(transformer, sentence, tokenizer_cn, tokenizer_en)

print(f'{"Input:":15s}: {sentence}')
print(f'{"Prediction":15s}: {predicted}')
print(f'{"Ground truth":15s}: {ground_truth}')
print("Bleu Score (1gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 1).item())
print("Bleu Score (2gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 2).item())
print("Bleu Score (3gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 3).item())
print("Bleu Score (4gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 4).item())
```

```
Input:          : 早上好，很高心见到你
Prediction      : You see the early morning, you see.
Ground truth    : Good Morning, nice to meet you
Bleu Score (1gram):  0.2857142984867096
Bleu Score (2gram):  0.0
Bleu Score (3gram):  0.0
Bleu Score (4gram):  0.0
```

```python
sentence = "祝您有个美好的一天"
ground_truth = 'Have a nice day'
predicted = translate(transformer, sentence, tokenizer_cn, tokenizer_en)

print(f'{"Input:":15s}: {sentence}')
print(f'{"Prediction":15s}: {predicted}')
print(f'{"Ground truth":15s}: {ground_truth}')
print("Bleu Score (1gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 1).item())
print("Bleu Score (2gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 2).item())
print("Bleu Score (3gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 3).item())
print("Bleu Score (4gram): ", bleu_score_func(predicted.lower(), ground_truth.lower(), 4).item())
```

```
Input:          : 祝您有个美好的一天
Prediction      : You have a good day.
Ground truth    : Have a nice day
Bleu Score (1gram):  0.4000000059604645
Bleu Score (2gram):  0.3162277638912201
Bleu Score (3gram):  0.0
Bleu Score (4gram):  0.0
```

Fig. 2: BLEU score

```
Epoch: 1, Train loss: 6.385, Val loss: 5.799, Val Acc: 0.157, Epoch time = 58.195s
(model saved)
Epoch: 2, Train loss: 5.542, Val loss: 5.481, Val Acc: 0.183, Epoch time = 58.579s
(model saved)
Epoch: 3, Train loss: 5.165, Val loss: 5.355, Val Acc: 0.188, Epoch time = 58.685s
(model saved)
Epoch: 4, Train loss: 4.894, Val loss: 5.313, Val Acc: 0.195, Epoch time = 58.728s
(model saved)
Epoch: 5, Train loss: 4.686, Val loss: 5.289, Val Acc: 0.201, Epoch time = 58.790s
(model saved)
Epoch: 6, Train loss: 4.520, Val loss: 5.301, Val Acc: 0.204, Epoch time = 58.818s
(model saved)
Epoch: 7, Train loss: 4.385, Val loss: 5.321, Val Acc: 0.204, Epoch time = 58.846s
Epoch: 8, Train loss: 4.278, Val loss: 5.352, Val Acc: 0.205, Epoch time = 58.994s
(model saved)
Epoch: 9, Train loss: 4.026, Val loss: 5.347, Val Acc: 0.212, Epoch time = 58.974s
(model saved)
Epoch: 10, Train loss: 3.947, Val loss: 5.372, Val Acc: 0.215, Epoch time = 58.957s
(model saved)
```

Fig. 3: Val Accuracy

2 In task-1

2.1 The structure of Transformer

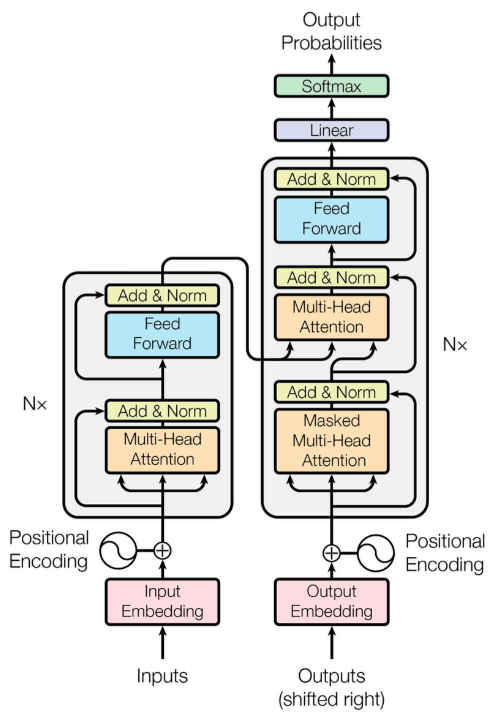

Fig. 4: The architecture of Transformer

在此次設計的架構裡，含有 Multi-Head Attention、Feed Forward Neural Network、Layer Normalization、Positional Encoding 等模組。

2.1.1 Transformer Encoder

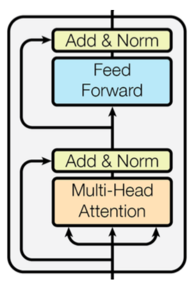

Fig. 5: Transformer Encoder

內部結構由兩個子層組成，分別是 Multi-head Self-Attention Sublayer 和 Position-wise Feed Forward Layer。Multi-head 讓模型可以學習到不同的特徵，而 Feed Forward Layer 則是對特徵進行線性轉換。

2.1.2 Transformer Decoder

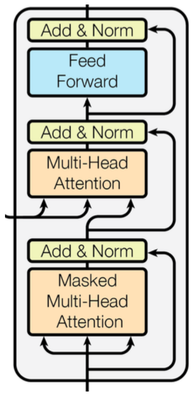

Fig. 6: Transformer Decoder

Decoder 和 Encoder 的結構類似，但在 Decoder 中多了一層。從下到上，先是 Masked Multi-head Self-Attention Sublayer，遮蔽未翻譯部分，讓模型只能看到已翻譯的詞，避免在生成的過程中「偷看」後面的詞。再來是 Multi-head Attention Sublayer，將翻譯中的句子和 Encoder 輸出的特徵進行注意力機制，使模型能在生成每個詞時充分參考原句的語意和結構。最後是 Position-wise Feed Forward Layer，對前面的特徵進行非線性變換，使模型能學習到更加複雜的表達。

```
[21]: EMB_SIZE = 128
NHEAD = 8
FFN_HID_DIM = 1024
NUM_ENCODER_LAYERS = 1
NUM_DECODER_LAYERS = 1
SRC_VOCAB_SIZE = tokenizer_cn.vocab_size
TGT_VOCAB_SIZE = tokenizer_en.vocab_size
DEVICE = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

transformer = Seq2SeqNetwork(NUM_ENCODER_LAYERS, NUM_DECODER_LAYERS, EMB_SIZE,
                             NHEAD, SRC_VOCAB_SIZE, TGT_VOCAB_SIZE, FFN_HID_DIM)

for p in transformer.parameters():
    if p.dim() > 1:
        nn.init.xavier_uniform_(p)

transformer = transformer.to(DEVICE)
param_transformer = sum(p.numel() for p in transformer.parameters())
print (f"The parameter size of transformer is {param_transformer/1000} k")
#  The parameter size of model should be less than 100M (100,000k) !!!
#  The parameter size of model should be less than 100M (100,000k) !!!
#  The parameter size of model should be less than 100M (100,000k) !!!

The parameter size of transformer is 10898.884 k
```

Fig. 7: Transformer Encoder

只用了各一層的 Encoder layer 和 Decoder layer，因為發現調大後，效果並沒有比較好，而且還跑比較久（參數量增加）。

2.2 Training strategy

用 ReduceLROnPlateau，根據模型表現來調整 learning rate。在指定 patience 沒有改善時，學習率就會乘以 factor 來衰減，避免提早結束訓練。而且還有提升收斂效果，減少過擬合的風險。

3 Improvement

1. 把 FFN_HID_DIM 調大，減少參數數量（跑得快些）。調整 NHEAD 可以讓 multi head attention 讓模型抓到不同 tokens 之間的關係。

4 Challenges I faced

1. 在 attention 的部分，很容易遇到維度不匹配的問題。預設的 attention 輸入是 (seq_length, batch_size, d_model)，但是在 decoder 的部分，我們需要把 encoder 的 output 和 decoder 的 output 做 attention，所以需要把 encoder 的 output 用 permute 轉置成 (batch_size, seq_length, d_model)。這樣才能做 attention。

   attention score shape: (n_head, batch_size, seq_length_Q, seq_length_K)
   假設 n_head = 8, batch_size = 64

   (a) future mask (decoder self-attention) shape:
   $(1, 1, 127, 127) \rightarrow (8, 64, 127, 127)$
   (b) padding mask (encoder-decoder attention) shape:
   $(1, 64, 1, 128) \rightarrow (8, 64, 128, 128)$
   (c) padding mask (decoder self-attention) shape:
   $(1, 64, 1, 127) \rightarrow (8, 64, 127, 127)$
   (d) padding mask (decoder cross-attention) shape:
   $(1, 64, 1, 128) \rightarrow (8, 64, 127, 128)$

2. 遇到預測是 nan 的問題。後面發現把 create_mask 加上去就可以解決。把 mask 加上去的原因是因為在訓練的時候，我們是用 mask 來過濾掉不需要的部分，所以在 translate 預測的時候也要加上去。encoder output 出來的 mask 也需要加上這些 mask。然後預設 PAD_IDX = 0，所以在 mask 的時候，要注意是把 PAD_IDX 的部分 mask 掉。

```
out: tensor([[[nan, nan, nan,  ..., nan, nan, nan]],

        [[nan, nan, nan,  ..., nan, nan, nan]],

        [[nan, nan, nan,  ..., nan, nan, nan]],

        ...,

        [[nan, nan, nan,  ..., nan, nan, nan]],

        [[nan, nan, nan,  ..., nan, nan, nan]],

        [[nan, nan, nan,  ..., nan, nan, nan]]], device='cuda:0',
       grad_fn=<NativeLayerNormBackward0>)
Input:          : 你好，欢迎来到中国
Prediction      :
Ground truth    : Hello, Welcome to China
Bleu Score (1gram):  0.0
Bleu Score (2gram):  0.0
Bleu Score (3gram):  0.0
Bleu Score (4gram):  0.0
```

5 References

1. transformer_baseline
2. Transformer from scratch using Pytorch
3. 加上 attention mask 之后 loss 出现 nan 问题的解决方案
4. Tutorial 6: Transformers and Multi-Head Attention
5. 29. Transformer