STAT8021 Assignment3 Report

Xia Hongyu

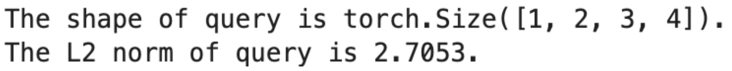
April 3, 2025

1 Introduction

Finetune a pre-trained Transformer model from hugging face. The dataset in this part is part of AG News.

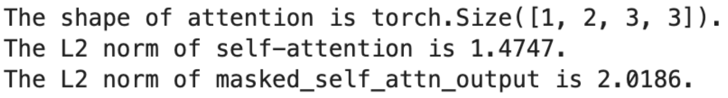
2 Understanding Transformer

2.1 (a)



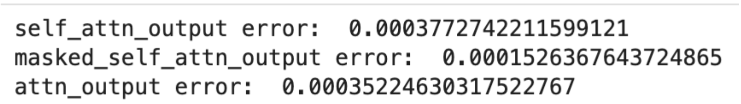
```
The shape of query is torch.Size([1, 2, 3, 4]).
The L2 norm of query is 2.7053.
```

2.2 (b)



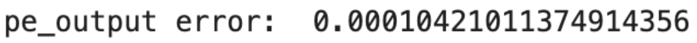
```
The shape of attention is torch.Size([1, 2, 3, 3]).
The L2 norm of self-attention is 1.4747.
The L2 norm of masked_self_attn_output is 2.0186.
```

2.3 (c)



```
self_attn_output error:  0.0003772742211599121
masked_self_attn_output error:  0.00015263676437248 65
attn_output error:  0.00035224630317522767
```

2.4 (d)

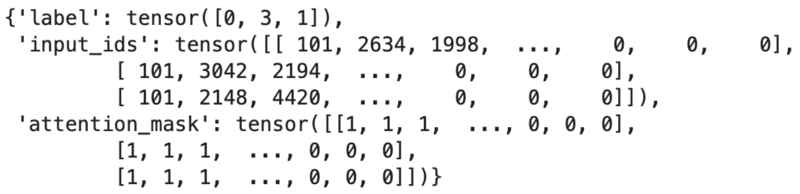


```
pe_output error:  0.00010421011374914356
```

3 Applying Transformer

3.1 (a)

The first 3 samples in the train set are:



```
{'label': tensor([0, 3, 1]),
 'input_ids': tensor([[ 101, 2634, 1998,  ...,    0,    0,    0],
        [ 101, 3042, 2194,  ...,    0,    0,    0],
        [ 101, 2148, 4420,  ...,    0,    0,    0]]),
 'attention_mask': tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0]])}
```

3.2 (b)

When using the pre-trained DistilBertForSequenceClassification, I selected a batch size of 8, a learning rate of 5e-5, and conducted three epochs of training. Below are the details and results for each epoch:

Epoch 1:	train acc = 0.7217	test acc = 0.8750
Epoch 2:	train acc = 0.9209	test acc = 0.8984
Epoch 3:	train acc = 0.9658	test acc = 0.8906

Time Consume: 25.5 min

3.3 (c)

Subsequently, the testing results on the four test samples are as follows:

News: In an exciting match last night, the Los Angeles Lakers defeated the Brooklyn Nets 115-110. Lakers' LeBron James made a comeback after missing several games due to injury and scored 25 points while teammate Anthony Davis added 28 points. Nets' star player Kevin Durant scored 32 points but couldn't lead his team to victory.
Result: Sports

News: Scientists have discovered a new species of dinosaur that roamed the earth 80 million years ago. The species, named Almatherium, was found in Uzbekistan and is believed to be an ancestor of the modern-day armadillo. The discovery sheds new light on the evolution of mammals and their relationship with dinosaurs.
Result: Sci/Tech

News: The United Nations has called for an immediate ceasefire in Yemen as the country faces a growing humanitarian crisis. The UN's special envoy for Yemen, Martin Griffiths, urged all parties to end the violence and engage in peace talks. The conflict has left millions of Yemenis at risk of famine and disease.
Result: World

News: Amazon has announced that it will be opening its first fulfillment center in New Zealand, creating more than 500 new jobs. The center will be located in Auckland and is expected to open in 2022. This move will allow Amazon to expand its operations in the region and improve delivery times for customers.
Result: Business

3.4 (d)

Finally, I chose to use RobertaForSequenceClassification along with its corresponding tokenizer, RobertaTokenizer, for testing. The testing parameters were the same as in question b. The final results are as follows:

Epoch 1:	train acc = 0.7285	test acc = 0.8438
Epoch 2:	train acc = 0.9131	test acc = 0.8672
Epoch 3:	train acc = 0.9492	test acc = 0.9141

Time Consume: 46.5 min

Comparation:

From the testing results, it can be observed that DistilBertForSequenceClassification and RobertaForSequenceClassification exhibit different strengths in terms of training and testing accuracy. DistilBERT has a shorter training time (25.5 minutes) and achieves a training accuracy (train acc) of 0.9658 at the third epoch. However, its improvement in testing accuracy is relatively limited, with a final test accuracy of 0.8906, indicating slightly weaker robustness on the test set. In contrast, Roberta has a longer training time (46.5 minutes) but demonstrates better optimization in testing accuracy, achieving a final test accuracy of 0.9141, which reflects stronger robustness and generalization. Overall, Roberta is more suitable for tasks requiring higher accuracy and robustness, while DistilBERT achieves a better trade-off between time and performance.

4 Transformer Pretraining and Finetuning Analysis

4.1 (a)

- The objective of the language modeling task is to train the model to predict the missing or next word in a sequence of text. By doing so, the model learns contextual representations of words, phrases, and sentences, enabling it to generate coherent and meaningful text or complete tasks based on textual inputs.

- Given the logits for the vocabulary {cat, dog, mat, ran, sat}:

  $$\text{logits} = [0.5, 0.1, 3.0, 0.2, 1.0]$$

  After we use the softmax function:

  $$P(i) = \frac{e^{\text{logit}_i}}{\sum_j e^{\text{logit}_j}}$$

  So:

  $$e^{0.5} \approx 1.6487, \quad e^{0.1} \approx 1.1052, \quad e^{3.0} \approx 20.0855, \quad e^{0.2} \approx 1.2214, \quad e^{1.0} \approx 2.7183$$

  The sum of exponentials is:

  $$\text{Sum} = 1.6487 + 1.1052 + 20.0855 + 1.2214 + 2.7183 \approx 26.7789$$

  The probability for the correct word ”mat” is:

  $$P(\text{mat}) = \frac{e^{3.0}}{\text{Sum}} = \frac{20.0855}{26.7789} \approx 0.7501$$

  The cross-entropy loss is computed as:

  $$\text{Loss} = -\log P(\text{mat}) \approx 0.2877$$

  So the result is:

  $$P(\text{mat}) \approx 0.7501, \quad \text{Loss} \approx 0.2877$$

4.2 (b)

The decoder-only architecture is well-suited for tasks like text generation due to its efficiency and ability to model sequential dependencies.

Compared to encoder-only models, it can perform better since it specializes in modeling sequences from left to right. However, it lacks bidirectional context understanding, making it less effective for classification tasks.

Compared to encoder-decoder models, it is simpler and more computationally efficient but less versatile, as encoder-decoder models are better suited for sequence-to-sequence tasks like translation and summarization.

4.3 (c)

- Size of a single attention score matrix:

  The size of each attention score matrix is $S_{\text{max}} \times S_{\text{max}}$.

  So total number of elements:

  $$S_{\text{max}}^2 = 16,384 \times 16,384 = 268,435,456$$

- Memory required to store a single matrix:

  Each element occupies 2 bytes, so the total memory for a single matrix is:

  $$\text{Memory} = 268,435,456 \times 2 = 536,870,912 \,\text{bytes} \approx 0.537 \,\text{GB}$$

- Considering all attention heads: There are $h = 48$ attention heads, each with its own attention score matrix. So the total memory required:

  $$\text{Total memory (GB)} = 0.537 \times 48 \approx 25.776 \,\text{GB}$$

4.4 (d)

- Total parameters for a single FFN layer while $H = 6144$:

  First layer: $H \to 4H$:

  $$H \cdot 4H + 4H = 151,019,520$$

  Second layer: $4H \to H$:

  $$4H \cdot H + H = 151,001,088$$

  Total parameters for a single layer:

  $$151,019,520 + 151,001,088 = 302,020,608$$

- Total parameters across all $L = 48$ layers:

  $$\text{Total parameters} = 48 \cdot 302,020,608 = 14,496,989,184$$

- Expressing in billions:

  $$\text{Total parameters (billion)} = \frac{14,496,989,184}{10^9} \approx 14.497 \,\text{billion}$$

4.5 (e)

4.5.1

- Total trainable parameters introduced by LoRA for a single matrix (W_Q or W_V):

  $$r \cdot H + H \cdot r = 2rH$$

- Total trainable parameters for both W_Q and W_V per layer:

$$\text{Parameters per layer} = 2 \cdot (2rH) = 4rH$$

- Total trainable parameters across all $L = 48$ layers:

$$\text{Total parameters} = L \cdot 4rH$$

- Substituting $L = 48$, $r = 8$, and $H = 6144$:

$$\text{Total parameters} = 48 \cdot 4 \cdot 8 \cdot 6144 = 9,437,184$$

- Expressing in millions:

$$\text{Total parameters (million)} = \frac{9,437,184}{10^6} \approx 9.437\,\text{million}$$

4.5.2

- Memory required per trainable parameter:

$$\text{Memory per parameter} = 2 \cdot 4\,\text{bytes} = 8\,\text{bytes}$$

- Total memory for all $9,437,184$ trainable parameters:

$$\text{Total memory} = 9,437,184 \cdot 8 = 75,497,472\,\text{bytes}$$

- Converting to MB:

$$\text{Total memory (MB)} = \frac{75,497,472}{10^6} \approx 75.497\,\text{MB}$$

4.5.3

- LoRA optimizer state memory (from part (e)(II)):

$$75.497\,\text{MB}$$

- FFN optimizer state memory:

$$\text{Total FFN parameters} \cdot \text{Memory per parameter} = 115,976,000,000\,\text{bytes}$$

  Converting to MB:

$$\text{FFN optimizer state memory (MB)} = \frac{115,976,000,000}{10^6} = 115,976\,\text{MB}$$

- Percentage comparison:

$$\text{Percentage} = \frac{\text{LoRA optimizer state memory}}{\text{FFN optimizer state memory}} \cdot 100$$

$$\text{Percentage} = \frac{75.497}{115,976} \cdot 100 \approx 0.065\%$$