Started on	Thursday, 28 November 2024, 4:17 PM
State	Finished
Completed on	Thursday, 28 November 2024, 4:31 PM
Time taken	13 mins 10 secs
Marks	15.00/20.00
Grade	**75.00** out of 100.00

Question 1

Complete

Mark 0.00 out of 1.00

What is the purpose of `math.sqrt(self.d_model)` in the following code?
```
def forward(self, x):
    return self.embedding(x) * math.sqrt(self.d_model)
```

- a. It normalizes the embedding values
- b. It ensures the embeddings have unit variance
- c. It scales the embeddings to maintain consistent variance across layers
- d. It improves the efficiency of the feed-forward network

Question 2

Complete

Mark 1.00 out of 1.00

What happens if the sequence length exceeds `seq_len` in the following snippet?
```
enc_num_padding_tokens = self.seq_len - len(enc_input_tokens) - 2
if enc_num_padding_tokens < 0:
    raise ValueError('Sentence is too long')
```

- a. Additional padding tokens are added to make up the length
- b. An error is raised, preventing processing of the batch
- c. The sequence length is dynamically adjusted
- d. The sentence is truncated to fit the allowed sequence length

Question 3

Complete

Mark 0.00 out of 1.00

What happens to the output of the encoder after all encoder blocks are processed?

- a. It is passed through a feed-forward layer before being sent to the decoder
- b. It is normalized and passed as input to the decoder
- c. Only the final token is passed to the decoder
- d. It is directly used as input for cross-attention in the decoder

Question 4

Complete

Mark 1.00 out of 1.00

What error would occur if the sequence length exceeds the defined `seq_len`?

- a. A `ValueError` will be raised during tokenization
- b. Attention scores will not converge
- c. Padding tokens will be ignored
- d. The model will generate tokens indefinitely

Question 5

Complete

Mark 1.00 out of 1.00

Which inputs are used for the query, key, and value in the encoder's self-attention mechanism?

- a. Query: `src_mask`, Key: `encoder_output`, Value: `decoder_input`
- b. Query: Decoder input, Key and Value: Encoder output
- c. Query, Key, and Value: Same encoder input sequence
- d. Query: `src_embeddings`, Key: `positional_encoding`, Value: `src_mask`

Question 6

Complete

Mark 0.00 out of 1.00

In the encoder, what is normalized by the `LayerNormalization` class?

- a. The positional encodings
- b. The input embeddings
- c. The output of each encoder block
- d. The attention weights

Question 7

Complete

Mark 1.00 out of 1.00

In the following code, which part ensures masking of invalid positions during attention?
`attention_scores = (query @ key.transpose(-2, -1)) / math.sqrt(d_k)
if mask is not None:
attention_scores.masked_fill_(mask == 0, -1e9)`

- a. `math.sqrt(d_k)`
- b. `attention_scores.masked_fill_(mask==0, -1e9)`
- c. None of the above
- d. `query @ key.transpose(-2, -1)`

Question 8

Complete

Mark 1.00 out of 1.00

Identify the sequence of operations in the encoder block from the following code snippet:
`def forward(self, x, src_mask):
x = self.residual_connections[0](x, lambda x: self.self_attention_block(x, x, x, src_mask))
x = self.residual_connections[1](x, self.feed_forward_block)
return x`

- a. Residual connection → Feed-forward block → Self-attention
- b. Self-attention → Residual connection → Feed-forward block
- c. Self-attention → Feed-forward block → Residual connection
- d. Self-attention with residual connection → Feed-forward block with residual connection

Question 9

Complete

Mark 1.00 out of 1.00

In the following method, why is `softmax` applied along the last dimension?
`attention_scores = attention_scores.softmax(dim=-1)`

- a. To increase the variance of attention scores
- b. To normalize the attention weights for each query
- c. To ensure padding tokens are ignored during training
- d. To scale the attention scores by the query-key dot product

Question 10

Complete

Mark 1.00 out of 1.00

In the multi-head attention implementation, what does the `w_q` parameter represent?

- a. A matrix that combines query, key, and value
- b. A learnable weight matrix for the query vectors
- c. A matrix that maps queries to values
- d. A mask to ignore padding tokens in queries

Question 11

Complete

Mark 1.00 out of 1.00

How does multi-head attention improve performance over single-head attention?

- a. By reducing computational complexity
- b. By normalizing attention scores more effectively
- c. By enabling attention across different subspaces of the input
- d. By computing attention over multiple queries

Question 12

Complete

Mark 1.00 out of 1.00

What is the purpose of the dropout layer in the multi-head attention mechanism?

- a. To mask out padding tokens
- b. To normalize the attention scores
- c. To prevent overfitting in attention weight computations
- d. To reduce the dimensionality of the attention output

Question 13

Complete

Mark 0.00 out of 1.00

What modification would allow the Transformer to handle longer sequences efficiently?

- a. Increasing the number of attention heads
- b. Reducing the number of encoder layers
- c. Using relative positional encodings
- d. Changing the optimizer to AdamW

Question 14

Complete

Mark 1.00 out of 1.00

What purpose do residual connections serve in the Transformer model, and where are they used in this code?

- a. They add positional information to embeddings
- b. They enforce weight sharing across layers
- c. They simplify the computation of attention scores
- d. They ensure gradient stability during training

Question 15

Complete

Mark 1.00 out of 1.00

What does the following snippet achieve in the positional encoding?
`div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-math.log(10000.0) / d_model))
pe[:, 0::2] = torch.sin(position * div_term)
pe[:, 1::2] = torch.cos(position * div_term)`

- a. Scales positional values to match input embeddings
- b. Computes learnable embeddings for token positions
- c. Encodes positional information using sine and cosine functions
- d. Normalizes the position vectors before feeding into the encoder

Question 16

Complete

Mark 1.00 out of 1.00

What is the purpose of the following block in `MultiHeadAttentionBlock`?
`query = query.view(query.shape[0], query.shape[1], self.h, self.d_k).transpose(1, 2)
key = key.view(key.shape[0], key.shape[1], self.h, self.d_k).transpose(1, 2)
value = value.view(value.shape[0], value.shape[1], self.h, self.d_k).transpose(1, 2)`

- a. Splits the query, key, and value into multiple attention heads
- b. Applies normalization to the query, key, and value tensors
- c. Prepares the tensors for the projection layer
- d. Combines query, key, and value for single-head attention

Question 17

Complete

Mark 1.00 out of 1.00

What does the dropout layer in `FeedForwardBlock` help prevent?

- a. Gradient explosion
- b. Excessive padding
- c. Overfitting
- d. Token misalignment

Question 18

Complete

Mark 0.00 out of 1.00

How does padding affect the attention mechanism in the encoder?

- a. Padding tokens are added after positional encoding
- b. Padding tokens are normalized using LayerNormalization
- c. Padding tokens are ignored using a mask
- d. Padding tokens are treated as regular tokens in attention computation

Question 19

Complete

Mark 1.00 out of 1.00

What is the primary role of the `PositionalEncoding` class?

- a. Normalizes token embeddings
- b. Provides positional information to the embeddings
- c. Initializes input embeddings with Xavier initialization
- d. Adds learnable embeddings for tokens

Question 20

Complete

Mark 1.00 out of 1.00

Which of the following steps is not included in the encoder's forward pass?

- a. Adding positional encodings
- b. Applying multi-head attention
- c. Applying cross-attention
- d. Generating token embeddings