Started on	Monday, 13 January 2025, 12:53 AM
State	Finished
Completed on	Monday, 13 January 2025, 1:09 AM
Time taken	16 mins 15 secs
Grade	**5.00** out of 5.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

In the transformer, we compute self-attention among all tokens relative to each other, potentially losing the __(i)__ . Here, __(ii)__ comes to the rescue

- i) embedding information, ii) positional encoding
- i) Semantic information, ii) Masking
- i) order information, ii) positional encoding ✔
- i) bag of words information, ii) embedding

Your answer is correct.

Within the framework of the transformer architecture, inherent order information in the input sequence is lost. However, this is effectively counteracted by positional encoding, which ensures that the model can still understand and consider the sequence's original order during processing.

The correct answer is:
i) order information, ii) positional encoding

Question **2**

Correct

Mark 1.00 out of 1.00

The computational complexity of attention weights calculation in a transformer is N^2. What is 'N' referring to?

- Dimensions of Q, K, V
- Embedding dimension

Embedding dimension

Batch size

Sequence length ✔

Your answer is correct.

Every term in a sequence must have an attention weight with every term. If there are N terms in a sequence (N=Sequence length), then it is required to compute N2 weights

The correct answer is:
Sequence length

Question **3**

Correct

Mark 1.00 out of 1.00

The table on the left, below, shows tokenized text data (as input) and the right table shows the shape of different objects used in the transformer architecture. Symbols have their usual meaning in the context of the transformer.

	ti = 1	ti = 2	ti = 3				ti = 100 (T)
S1	I	am	playing	cricket	...	...	....
S2	...	...	...	..			...
S3	...	...	...	..			...
...							
...							
...							
...							
S100							
...							
...							
...							
...							

Batch size = N (Say, 100)

Object	Shape
q_i , k_i	d_k
v_i	d_v
x_i	d_{model}
$W^{(Q)}, W^{(K)}$	$d_{model} \times d_k$
$W^{(V)}$	$d_{model} \times d_v$

What will the shape of the output tensor be after passing through the multi-head attention of an encoder block with 6 attention heads? Consider the batch size 'N' and original sequence length 'T'.

- (N, T, $6d_v$)
- (N, T, d_v)
- (N, T, d_{model}) ✔

Your answer is correct.

In the transformer's multi-head self-attention mechanism, the input tensor's sequence length remains unchanged, and the model dimension remains the same as the input dimension. The number of attention heads (6 in this case) operates independently on the input, generating '6' sets of attention-weighted outputs. These '6' sets of attention outputs are then concatenated and linearly transformed to yield the final output. Therefore, the shape of the output tensor is 'N' batches, maintaining the original sequence length (T), and having the same model dimension as the input i.e. d_{model}

The correct answer is:
(N, T, d_{model})

Question **4**

Correct

Mark 1.00 out of 1.00

Which of the following is a Natural Language Understanding (Predictive AI) Task?

i) Sentiment Classification
ii) Neural Machine Translation
iii) Sentence Completion
iv) Topic Modeling

- Only (i)
- Only (iv)
- (ii) and (iii)
- (i) and (iv) ✔

Your answer is correct.

In NLU tasks, the goal is to understand language and perform a discriminative activity such as classification. It is usually a sequence-to-vector model. Sentiment Classification and Topic Modeling are examples of NLU tasks. NMT and Sentence completion are a Seq-to-Seq task which comes under the regime of Natural Language Generation (also called GenAI task)

The correct answer is:
(i) and (iv)

Question **5**

Correct

Mark 1.00 out of 1.00

In which of the situations would an encoder only model be a great fit?

- For Neural Machine translation
- For creating an open-domain chatbot
- For building a spam detector ✔
- For building a lyrics writer

Your answer is correct.

Encoder-only models are suitable for language understanding tasks such as classification and spam detection is a text classification task. All other tasks mentioned in the options are generative tasks that need a decoder or both encoder + decoder type of model.

The correct answer is:
For building a spam detector