**SWS3009A Robotics and Deep Learning**

**Deep Learning Lab 2 Answer Book**

**SUBMISSION DEADLINE: Wednesday 3 JULY 2024, 11.59 pm**

Team Member 1 Name:WEN SIJIE

Team Member 2 Name:YANG RUNKANG

Team Member 3 Name (if any):

**Marks:** \_\_\_\_\_\_\_\_ / 3

**Please save as PDF before submitting to Canvas.**

**Question 1 Answer:**

Transformers and LSTMs differ significantly in their architecture and attention mechanisms. The core component of transformers is the self-attention mechanism, which allows each token in a sentence to attend to every other token, effectively capturing long-range dependencies and enabling transformers to process entire sentences simultaneously. In contrast, LSTMs process sequences step-by-step, maintaining a hidden state that evolves over time, handling dependencies within a fixed window of tokens and relying on the hidden state to carry contextual information from previous tokens. Regarding parallelization and efficiency, transformers leverage parallel computation by processing all tokens in a sentence at once, which is more efficient for training on modern hardware as entire sentences are fed into the model simultaneously. Conversely, LSTMs process tokens sequentially, limiting parallelization and making them slower to train, especially on long sequences. In terms of contextual understanding, the self-attention mechanism in transformers allows them to capture relationships between distant tokens, leading to a better understanding of context within a full sentence. While LSTMs can capture some long-term dependencies, their effectiveness diminishes with very long sequences due to the vanishing gradient problem and a limited look-back window.

**Question 2 Answer:**

The sentences used to train transformers must be of fixed length due to the architectural requirements of most neural network models, including transformers. Neural networks, in general, require a predefined and consistent input shape and size for training. Transformers apply self-attention across the entire sequence, and handling variable-length sentences directly would require dynamic adjustment of weights and memory allocations within the network, complicating the training process. To manage this, input sentences are typically padded or truncated to a consistent length, ensuring that each input tensor to the transformer has the same shape, which simplifies the computational requirements and makes the model training more efficient and manageable.

**Question 3 Answer:**

model\_name: This specifies the name or path of the pretrained model configuration from which to load the settings. Here, it's gpt2, indicating we are using the configuration of the GPT-2 model.

vocab\_size: This sets the number of unique tokens that the model can use. It's set to the length of the tokenizer, which means the model will have as many tokens as were found in the tokenizer.

n\_ctx (context size or sequence length): This is the maximum length of the input sequences the model can handle. Here, it is set to max\_length, specifying that the model should process sequences up to 30 tokens long.

bos\_token\_id (beginning-of-sentence token ID): This specifies the token that denotes the start of a sentence.

eos\_token\_id (end-of-sentence token ID): This indicates the token that denotes the end of a sentence.

**Question 4 Answer:**

Comparing the texts generated from a transformer trained from scratch versus one using pretrained GPT2 weights typically reveals significant differences in quality. The model using pretrained GPT2 weights generally produces text that is more coherent, contextually appropriate, and contains fewer syntactical or semantic errors. This is because the pretrained model has been trained on a vast amount of text and has learned robust language patterns and nuances that are not easily captured in a model trained from scratch on a smaller dataset. Pretrained models also tend to generate text with fewer "non-English" words or nonsensical sequences, reflecting their training on large, diverse language corpora.