**SWS3009A Robotics and Deep Learning**

**Deep Learning Lab 1 Answer Book**

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

Team Member 1 Name: YANG RUNKANG

Team Member 2 Name: WEN SIJIE

Team Member 3 Name (if any):

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

Please save as PDF before submitting to Canvas.

**Question 1 Answer:**

i) What an embedding layer does: An embedding layer is used in neural networks to convert sparse, high-dimensional data (like integer representations of words) into a lower-dimensional, dense, and continuous vector space. Each word or token is mapped to a vector of real numbers which capture semantic relationships between words. This makes it easier for the model to process and learn patterns from the data.

ii) Why we cannot just feed the integers from the tokenizer directly to the LSTM: Simply feeding the integers (which are categorical in nature) directly into an LSTM would not effectively capture the semantic relationships between different words. Integers as inputs could imply a false ordinal relationship (where higher numbers represent "more" or "better") that doesn't exist in natural language. Additionally, using integers would increase the model's complexity due to the large input space (one input node per unique word), making it inefficient and harder to train. Embedding layers solve these issues by representing words in a more meaningful, dense format that an LSTM can more effectively learn from.

**Question 2 Answer:**

(Fill answer here)

Transformers: Transformers use an attention mechanism that allows each position in the sequence to attend to every other position in a single operation. This means that transformers can handle entire sequences at once and model relationships between tokens regardless of their position. Therefore, they do not require the input to be broken down into smaller sequences because they can effectively capture the context of the whole sequence in parallel.

LSTMs: Long Short-Term Memory networks (LSTMs), on the other hand, process data sequentially and rely on their stateful nature to capture information from the input over time. To train an LSTM to predict the next word, it is essential to feed it sequences of a fixed length so that it learns to predict the next output based on a specific number of previous tokens. This training process helps the LSTM understand the temporal dependencies between tokens in a sequence

**Question 3 Answer:**

i. Why we can't have one output instead of one-hot encoding:

Using a single output neuron that directly predicts the index of the next word would turn the problem into a regression task, where the model tries to output a continuous value that is closest to the target word index. However, language modeling is inherently a classification problem, where each output represents a distinct class (word). One-hot encoding transforms this classification problem into a format suitable for neural networks, where each word is represented as a vector with all zeros except for a one at the index of the word. This representation distinctly separates each word as a different class and allows the network to learn to predict the probability distribution over all possible words, which is much more suitable for handling the discrete nature of language.

ii. Why use softmax and categorical cross entropy:

Softmax Activation: The softmax function is used as the activation function in the output layer of a classifier to convert logits (raw prediction scores) into probabilities by normalizing them into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. This is crucial in multi-class classification tasks like language modeling because it allows the model to output a probability for each class, with all probabilities summing up to one.

Categorical Cross-Entropy Loss: This loss function is used when there are two or more label classes. It measures the performance of a classification model whose output is a probability value between 0 and 1. Categorical cross-entropy loss compares the predicted probability distribution with the actual distribution (the true labels in one-hot encoded form) and punishes the probabilities diverging from the actual labels. It is highly effective for training models in tasks where each instance can only belong to one class, making it the standard choice for multi-class classification problems.