**SWS3009A Robotics and Deep Learning**

**Deep Learning Lab 1 Answer Book**

**SUBMISSION DEADLINE: Saturday 8 July 2023, 11.59 pm**

Team Member 1 Name:

Zheng Qinghe

Team Member 2 Name:

Zhao Qianyi

Team Member 3 Name (if any):

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

Please save as PDF before submitting to Canvas.

**Question 1 Answer:**

i) An embedding layer is a fundamental component in natural language processing (NLP) and deep learning models. Its purpose is to transform categorical variables, such as words or tokens, into continuous vector representations (embeddings) that capture their semantic meaning and relationships. The embedding layer maps each input token to a dense vector of fixed size, where similar tokens are closer together in the embedding space. This mapping allows the model to learn meaningful representations of the words and capture their contextual information.

The embedding layer is typically initialized randomly and gets updated during the training process, as the model learns to optimize the embeddings for the specific task at hand. By learning these representations from the data, the model can leverage the semantic relationships between words, even for words that did not appear in the training set.

ii)

1. The integers produced by the tokenizer are categorical identifiers and do not carry any inherent meaning or similarity information. They are arbitrary and lack semantic significance.

2. Neural network models, such as LSTMs, require a fixed input size. However, text sequences often have varying lengths. If we were to directly feed the integer tokens, we would need to either truncate or pad the sequences to a fixed length. Truncation can result in the loss of important information, while padding introduces noise and inefficiency.

3. The integers do not inherently capture the semantic relationships between the tokens. Neural networks, including LSTMs, rely on learned weights to establish relationships between input features. Feeding raw integers would not provide meaningful information to learn these relationships effectively.

**Question 2 Answer:**

This is because Transformer is designed to process the entire input sequence simultaneously rather than sequentially.

Transformers use a self-attention mechanism to capture dependencies between different positions in the input sequence. This allows the model to attend to all positions in the sequence at once and capture global dependencies, without the need for sequential processing or predicting tokens one by one.

Besides, Transformers can process the entire input sequence in parallel. The self-attention mechanism enables direct interactions between all positions, allowing for efficient parallel computation. This parallel processing capability makes it unnecessary to chop the tokens into fixed-length groups for prediction.

On the contrary, LSTM and other recurrent neural network (RNN) models are inherently sequential models. They process sequences step by step, and the hidden state depends on the previous time steps. Therefore, LSTM models require fixed-length input sequences during training.

**Question 3 Answer:**

1. The one-hot approach is used because it allows the network to treat the next word prediction as a multiclass classification problem.

If we were to use a single output neuron and use the target value as the index of the next word, it would transform the problem into a regression task. However, in the case of next word prediction, we want to predict discrete words rather than continuous values. The one-hot encoding provides a clear distinction among the different words in the vocabulary and enables the model to learn the categorical relationships between them.

1. We use softmax and categorical cross entropy loss because they are suitable for multiclass classification tasks like next word prediction.

The softmax function takes a vector of real values and normalizes them into a probability distribution, ensuring that the predicted probabilities sum up to 1. In the context of next word prediction, softmax ensures that the model's outputs represent probabilities for each word in the vocabulary. It helps in selecting the most likely next word based on the predicted probabilities.

Categorical cross entropy measures the dissimilarity between the predicted probability distribution and the true target distribution. In the case of next word prediction, the true target distribution is represented as a one-hot encoded vector. By comparing the predicted probabilities with the true target distribution, the categorical cross-entropy loss quantifies the discrepancy between the predicted and actual probabilities of the next word. Minimizing this loss encourages the model to improve its ability to predict the correct next word.

These choices allow the model to effectively learn the patterns and dependencies in the language and generate accurate predictions for the next word.