Task 1: Conceptual Questions

Answer the following questions in 2–4 sentences each:

1. What is the difference between RNN and LSTM?

ANS:-

|  |  |  |
| --- | --- | --- |
| Parameter | RNN | LSTM |
| Handling Long Sequences | Struggles to maintain long-term dependencies due to vanishing gradients | Excels in capturing long-term dependencies |
| Training Time | Fast but less accurate on complex data | Slower due to multiple gates and memory operations |
| Use Cases | stock prices | text generation |

1. What is the vanishing gradient problem, and how does LSTM solve it?

ANS:-

It occurs during backpropagation When updating weights and biases using gradient descent if the gradients are too small the updated to weights and biases become insufficient slowing down or even stopping learning

LSTM solve this by using a memory cell and three gates—input, forget, and output—that regulate information flow. These gates allow LSTMs to selectively retain or discard information, maintaining stable gradients and enabling the learning of long-term dependencies.

1. Explain the purpose of the Encoder-Decoder architecture.

ANS:-

The Encoder-Decoder architecture is designed to handle sequence-to-sequence tasks such as machine translation or text summarization.

Encoder: The encoder takes the input data like a sentence and processes each word one by one then creates a single and fixed-size summary of the entire input called a context vector. Decoder: The decoder takes the context vector and begins to produce the output one step at a time.

4. In a sequence-to-sequence model, what are the roles of the encoder

and decoder?

ANS:-

the encoder processes the input sequence to create a fixed-size representation like a context vector. then the decoder uses that vector to generate the output sequence step by step. It often uses previous outputs to help with the next one.

5. How is attention different from a basic encoder-decoder model?

ANS:-

In a basic encoder-decoder architecture the encoder processes the entire input sequence and compresses it into a single fixed-size context vector. The decoder then relies solely on this context vector to generate the output sequence.Attention improves this by allowing the decoder to access all encoder hidden states at each decoding step focusing on the most relevant parts of the input.

Task 2: Sequence-to-Sequence Data Flow

Draw or describe the data flow in an encoder-decoder model using

RNN/LSTM.

Clearly label:

● Input sequence: The English sentence “hello how are you” is cleaned (lowercase, no punctuation), tokenized (e.g., [2, 50, 30, 10, 3] for , “hello,” “how,” “are you,” ), and padded to a fixed length (e.g., 6). Each token is turned into a 256-dimensional vector via an embedding layer, giving a sequence of vectors [e\_2, e\_50, e\_30, e\_10, e\_3, e\_0].

● Hidden states: The encoder LSTM processes these embeddings one by one. For each token, it updates a hidden state h\_t and cell state c\_t (both 512-dimensional, per your lstm\_units). By the end, we have a sequence of hidden states [h\_1, h\_2, h\_3, h\_4, h\_5] and final states h\_T, c\_T. Your code also applies self-attention, comparing hidden states to create a weighted encoder context.

● Context vector: Instead of just using h\_T, attention creates a dynamic context vector for each decoder step. The decoder’s current hidden state h\_dec\_t is compared (via dot product) to all encoder hidden states [h\_1, h\_2, ...], producing attention weights. These weights are used to combine the encoder’s hidden states into a context vector, focusing on relevant input words (e.g., “hello” for “bonjour”).

● Output sequence: The decoder starts with a token and the encoder’s h\_T, c\_T. For each step, it embeds the current token, processes it with an LSTM to get h\_dec\_t, computes a context vector via attention, concatenates them, and predicts the next French word (e.g., “bonjour”) using a dense layer. This repeats until or a max length, producing a sequence like “bonjour comment vas-tu”.

Task 8: Model Performance Discussion

1. What are the challenges in training sequence-to-sequence models?

ANS:-

It's hard because long sentences can be forgotten, and the model needs lots of data. It may also struggle to align words correctly between languages.

1. What does a “bad” translation look like? Why might it happen?

ANS:-

A bad translation might repeat words, miss important parts, or make no sense. This happens when the model doesn't learn properly or guesses wrong.

1. How can the model be improved further?

ANS:-

Use attention to help focus on the input, add more data, try beam search for better output, or switch to a Transformer model for better results.