**TASK-1**

**1. What is the difference between RNN and LSTM?**

* **RNN (Recurrent Neural Network)**:
  + Processes sequential data by maintaining a hidden state that gets updated at each time step.
  + Suffers from **vanishing/exploding gradients**, making it hard to learn long-term dependencies.
* **LSTM (Long Short-Term Memory)**:
  + A special type of RNN designed to learn long-term dependencies more effectively.
  + Uses **gates** (input, forget, output) and a **cell state** to control the flow of information.
  + Better at preserving information over longer sequences than basic RNNs.

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

* **Vanishing Gradient Problem**:
  + During backpropagation in deep or recurrent networks, gradients can become very small.
  + This causes the model to stop learning long-term dependencies, as updates shrink exponentially over time steps.
* **How LSTM Solves It**:
  + LSTM uses a **cell state** and **gating mechanisms** that allow gradients to flow more effectively through time.
  + The **forget gate** controls what information is discarded.
  + This structure helps **preserve gradients**, enabling learning over longer time spans.

**3. Explain the purpose of the Encoder-Decoder architecture.**

* The **Encoder-Decoder architecture** is designed for **sequence-to-sequence tasks** (e.g., machine translation, text summarization).
* **Purpose**:
  + The **encoder** processes the input sequence and compresses it into a **context vector** (a fixed-length summary).
  + The **decoder** uses this context vector to generate the output sequence step-by-step.

**4. In a sequence-to-sequence model, what are the roles of the encoder and decoder?**

* **Encoder**:
  + Takes the input sequence (e.g., a sentence in English).
  + Converts it into a fixed-length vector representation (context vector).
* **Decoder**:
  + Takes the context vector and generates the target sequence (e.g., translated sentence in French), one token at a time.
  + Often uses its own previous output as input at the next step.

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

* **Basic Encoder-Decoder**:
  + Uses a single context vector (from the final encoder state) to represent the entire input sequence.
  + Struggles with long sequences because information gets compressed too much.
* **Attention Mechanism**:
  + Allows the decoder to **access all encoder hidden states** at each decoding step.
  + Computes a **weighted sum** of these states based on how relevant each input token is to the current decoding step.
  + Helps the model **focus** on specific parts of the input sequence, improving performance on longer or more complex sequences.

**TASK-2**

**QUESTION - Draw or describe the data flow in an encoder-decoder model using RNN/LSTM.**

Encoder Decoder

┌─────────────────┐ ┌───────────────────────┐

│ Input Sequence │ │ Output Sequence │

│ (x1, x2, x3) │ │ (y1, y2, y3, ...) │

└─────────────────┘ └───────────────────────┘

↓ ↑

┌─────────────┐ ┌──────────────┐

│ h1 = RNN(x1)│ │ Decoder RNN │

└─────────────┘ └──────────────┘

↓ ↑

┌─────────────┐ ┌──────────────────────┐

│ h2 = RNN(x2, h1)│ ←─────── │ y1 = RNN(y0, context)│

└─────────────┘ context └──────────────────────┘

↓ ↑

┌─────────────┐ ┌──────────────────────┐

│ h3 = RNN(x3, h2)│ →──────── │ y2 = RNN(y1, context)│

└─────────────┘ └──────────────────────┘

↓

Context Vector = h\_last = h3

1. **Input Sequence**:
   * A sequence of tokens: x1, x2, x3, ..., xn
   * Fed one at a time into the encoder (RNN/LSTM).
2. **Hidden States (Encoder)**:
   * At each step: h1 = RNN(x1), h2 = RNN(x2, h1), ..., hn = RNN(xn, h\_{n-1})
3. **Context Vector**:
   * Final hidden state of the encoder: hn
   * Represents the summary of the entire input sequence.
   * Passed to the decoder as initial input.
4. **Output Sequence**:
   * Decoder uses the context vector and previous outputs to generate: y1, y2, y3, ..., ym
   * Each output token yi is generated by the decoder RNN conditioned on y\_{i-1} and the context.

**TASK-8**

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.