Step 1: Data

1) Number of data samples: 40000

2) Problem description: The dataset is for a machine translation task, translating English text to French.

3) Total words in the dataset: English - 133371, French - 157550

4) Missing information: None

5) Label description: The labels are French translations of the English texts.

6) Data split: Training - 80.0%, Validation - 10.0%, Testing - 10.0%

7) Data preprocessing:

For the training dataset, the preprocessing steps include:

- Tokenizing the English and French texts into characters.

- Converting characters to integer indices.

- One-hot encoding the characters.

- Padding sequences to ensure uniform length.

Step 2: Model Evaluation Metric

The evaluation metric which I used is 'accuracy', which is a common metric for classification tasks.

Model	Accuracy
LSTM (1 layer)	0.9402
LSTM (2 layers)	0.9332
LSTM (3 layers)	0.9340

Step 3: Objective - Loss Function

The loss function which I used is 'categorical_crossentropy', which is suitable for classification tasks where the output is a probability distribution across different classes. In the context of language translation, each output token is treated as a class, and the model predicts the probability of each token at each position in the sequence.

Step 4: Optimization - Optimizer Selection

I have used 'RMSprop' as the optimizer. RMSprop is an adaptive learning rate method, which means it adjusts the learning rate during training. It's particularly effective for recurrent neural networks (RNNs) like LSTM because it can navigate the complex landscapes of these models more effectively than simple optimizers like stochastic gradient descent (SGD).

Step 5: Model Selection

To determine the best-performing model, I compared the training and validation loss across models with different numbers of LSTM layers (1, 2, and 3 layers as per your code). The model with the lowest validation loss (and preferably a low training loss) would typically be considered the best. The learning rate is an important hyperparameter which I have set to 0.001 for model comparison.

Model	0.1	0.01	0.001
LSTM (1 layer)	0.8824	0.9402	0.9208
LSTM (2 layers)	0.8991	0.9332	0.9256
LSTM (3 layers)	0.8994	0.9340	0.9203

The best performing model was → LSTM (1 layer) with 0.01 Learning Rate.

Step 6: Model Performance

A & B → I have included the performance plots for training and validation loss across different models in the jupyter notebook. I have also included some sample translation results in the jupyter notebook itself.