Assignment: Sequence Modelling with LSTM and Encoder–Decoder Architectures

Bootcamp on Natural Language Processing

August 15, 2025

Objective

The objective of this assignment is to apply **Long Short-Term Memory (LSTM)** networks, either in a simple sequence model or within an **Encoder–Decoder** framework, to solve a practical sequence-based learning problem. Participants are expected to implement their solution in **Google Colab** using **PyTorch** (or optionally Keras/TensorFlow), train and evaluate the model, and present their findings.

Prerequisites

Participants should already understand:

- The functioning of Recurrent Neural Networks (RNNs) and their limitations.
- The role of LSTM gates (input, forget, output) and cell state.
- The architecture and working of Encoder–Decoder models for sequence-to-sequence tasks.
- Basic PyTorch (or Keras) workflow for model definition, training, and evaluation.

Task Description

Select a sequence learning task from one of the following domains:

- **Text:** e.g., text generation, next-word prediction, sentiment classification, or sequence labelling (NER, POS tagging).
- **Translation:** e.g., English–French or English–Hindi translation using a small parallel corpus.
- **Time Series:** e.g., stock price prediction, weather forecasting, or sensor signal prediction.
- **Dialogue:** e.g., chatbot-style response generation on a toy dataset.

Implement one of the following:

a. An **LSTM-based** sequence model

b. An **Encoder–Decoder** model where both encoder and decoder use LSTM cells.

Technical Requirements

1. **Dataset:** Choose a dataset that is small enough to train within Google Colab constraints yet rich enough to demonstrate sequence dependencies. Document the source and any preprocessing steps.

2. **Model Architecture:**

   - Define the embedding layer (if applicable).
   - For LSTM: specify the number of layers, hidden units, and directionality.
   - For Encoder–Decoder: specify encoder configuration, decoder configuration, and method of passing context vectors.

3. **Training:** Train the model for sufficient epochs, ensuring loss convergence. Use appropriate optimisation and regularisation techniques.

4. **Evaluation:** Select evaluation metrics suited to the task (e.g., accuracy, BLEU score, RMSE).

5. **Analysis:** Provide an analysis of:

   - Training and validation performance over epochs.
   - Strengths and weaknesses of the model on the given task.
   - Impact of LSTM architecture choices (e.g., hidden size, number of layers) on results.

Deliverables

Participants must submit:

1. A **Google Colab notebook** containing:

   - Data loading and preprocessing steps.
   - Model definition and explanation.
   - Training loop with loss tracking.
   - Evaluation code and results.

2. A brief **written report** (2–4 pages) summarising:

   - Problem statement and dataset description.
   - Model architecture and hyperparameters.
   - Results and analysis.
   - Potential improvements and next steps.

Evaluation Criteria

- **Functionality:** Correct and complete implementation of the LSTM or LSTM-based Encoder–Decoder (40%).
- **Clarity:** Well-documented code and clear explanation of model architecture (20%).
- **Performance:** Quality of results according to chosen metrics (20%).
- **Analysis:** Depth of insight in the written report (15%).
- **Reproducibility:** Notebook runs successfully in Google Colab without modifications (5%).