**Translation Model – Project Documentation**

**Project Overview**

This project implements a neural machine translation (NMT) system using a character-level sequence-to-sequence (seq2seq) architecture with Long Short-Term Memory (LSTM) layers in Keras. The goal is to translate sentences from a source language (such as English) to a target language (such as Arabic) using supervised learning on a parallel corpus.

The model learns to map input sequences to corresponding output sequences by being trained on pairs of sentences. Once trained, it can generate translations for unseen input sentences using the learned patterns.

**Data Pipeline and Preparation**

**Dataset**

The input dataset is a plain text file where each line contains a pair of sentences separated by a tab character. The first part is the source sentence, and the second part is the target sentence. A third field may be present but is not used.

**Preprocessing**

* Each line is split into source and target components.
* The target sentence is augmented with a start-of-sequence token (\t) and an end-of-sequence token (\n).
* A set of unique characters is extracted from both source and target texts to create vocabularies.
* Each character is mapped to a unique index for conversion to numerical format.
* The number of samples used is limited to 10,000 to manage computational requirements.

**Character-Level Modeling**

This project uses character-level modeling instead of word-level tokenization. Each character is treated as a separate unit, which reduces vocabulary size and handles rare or unknown words more effectively.

* All characters are represented using one-hot encoding.
* Input and output sequences are padded to uniform lengths to allow batch processing.
* Three main data structures are created: encoder input data, decoder input data, and decoder target data.

**Model Architecture**

**Encoder**

* Accepts sequences of one-hot encoded characters from the source language.
* Processes the sequence through an LSTM layer to produce two internal state vectors: the hidden state and the cell state.
* These state vectors represent the context of the input sentence.

**Decoder**

* Initializes with the encoder’s final states.
* Uses an LSTM layer to generate one character at a time in the target language.
* A dense layer with softmax activation predicts the next character based on the current decoder state and input.

**Teacher Forcing**

During training, the decoder is provided with the correct previous character from the target sequence instead of its own previous prediction. This approach helps the model converge faster and improves accuracy.

**Training**

* The model is compiled using the RMSprop optimizer and categorical crossentropy loss.
* Training is conducted over 100 epochs with a batch size of 64.
* A validation split of 20% is used to monitor performance on unseen data.
* The trained model is saved in the Keras format for future inference.

**Inference and Prediction**

Inference requires slightly different architecture than training:

* The encoder model is used to convert input sequences into context vectors.
* A separate decoder model generates the output sequence one character at a time, using the encoder’s output and the previously generated characters.

Decoding follows a greedy strategy:

* Start with the start-of-sequence token.
* Predict the next character with the highest probability.
* Feed that character back into the decoder for the next prediction.
* Continue until the end-of-sequence token is generated or a maximum length is reached.

**Evaluation**

The model is evaluated by generating translations for 20 sample input sentences from the training set. Each predicted sentence is compared with its corresponding input to qualitatively assess translation accuracy.

**Technical Concepts**

* **Sequence-to-Sequence (Seq2Seq):** A model architecture that maps input sequences to output sequences using encoder-decoder networks.
* **LSTM (Long Short-Term Memory):** A type of recurrent neural network suitable for handling sequences with long-range dependencies.
* **Character-Level Modeling:** Each character is processed independently, allowing fine-grained control and reducing vocabulary size.
* **One-Hot Encoding:** A method of representing categorical data (characters) as binary vectors.
* **Teacher Forcing:** A training technique where the correct output at time t-1 is fed into the decoder at time t.
* **Greedy Decoding:** A simple decoding method that always selects the most probable next character.
* **State Propagation:** The encoder’s output states are used to initialize the decoder’s internal state for context retention.

**Workflow Summary**

1. Load and preprocess the parallel text data.
2. Tokenize characters and build one-hot encoded data tensors.
3. Define and compile the encoder-decoder model.
4. Train the model with teacher forcing and save it.
5. Use separate encoder and decoder models during inference.
6. Translate new sentences using the decoding function.