**Aim:** Write a program to build a Neural Machine Translation (NMT) model using an encoder-decoder architecture with attention mechanism to translate sentences from one language to another (e.g., English to French).

**Description:**

Neural Machine Translation is an end-to-end learning approach for automated translation using deep neural networks. Unlike traditional statistical methods, NMT models learn to map input sequences (sentences) in a source language to output sequences in a target language, capturing complex patterns and context. This project uses sequence-to-sequence models with attention mechanism to improve translation quality.

**Objective:**

 To implement a sequence-to-sequence (Seq2Seq) NMT model with an attention mechanism.

 To train the model on a bilingual dataset (e.g., English-French).

 To evaluate the translation performance on test sentences.

**Steps Overview:**

 Data Preparation:

* Load and preprocess bilingual sentence pairs.
* Tokenize and pad sequences.

 Model Architecture:

* Build an encoder using RNN/LSTM/GRU layers.
* Build a decoder with attention mechanism.

 Training:

* Compile the model with appropriate loss and optimizer.
* Train on parallel corpus data.

 Evaluation:

* Translate test sentences.
* Evaluate output quality using BLEU score or manually.

**Implementation:**

*import numpy as np*

*import tensorflow as tf*

*from tensorflow.keras.models import Model*

*from tensorflow.keras.layers import Input, LSTM, Dense, Embedding*

*from tensorflow.keras.preprocessing.text import Tokenizer*

*from tensorflow.keras.preprocessing.sequence import pad\_sequences*

*# Sample data*

*eng\_sentences = ['hello', 'how are you', 'thank you']*

*fra\_sentences = ['bonjour', 'comment ça va', 'merci']*

*# Tokenization*

*eng\_tokenizer = Tokenizer()*

*fra\_tokenizer = Tokenizer()*

*eng\_tokenizer.fit\_on\_texts(eng\_sentences)*

*fra\_tokenizer.fit\_on\_texts(fra\_sentences)*

*eng\_seqs = eng\_tokenizer.texts\_to\_sequences(eng\_sentences)*

*fra\_seqs = fra\_tokenizer.texts\_to\_sequences(fra\_sentences)*

*eng\_padded = pad\_sequences(eng\_seqs, padding='post')*

*fra\_padded = pad\_sequences(fra\_seqs, padding='post')*

*# Model parameters*

*embedding\_dim = 64*

*units = 128*

*input\_vocab\_size = len(eng\_tokenizer.word\_index) + 1*

*target\_vocab\_size = len(fra\_tokenizer.word\_index) + 1*

*# Encoder*

*encoder\_inputs = Input(shape=(None,))*

*enc\_emb = Embedding(input\_vocab\_size, embedding\_dim)(encoder\_inputs)*

*encoder\_lstm = LSTM(units, return\_state=True)*

*encoder\_outputs, state\_h, state\_c = encoder\_lstm(enc\_emb)*

*# Decoder*

*decoder\_inputs = Input(shape=(None,))*

*dec\_emb = Embedding(target\_vocab\_size, embedding\_dim)(decoder\_inputs)*

*decoder\_lstm = LSTM(units, return\_sequences=True, return\_state=True)*

*decoder\_outputs, \_, \_ = decoder\_lstm(dec\_emb, initial\_state=[state\_h, state\_c])*

*decoder\_dense = Dense(target\_vocab\_size, activation='softmax')*

*decoder\_outputs = decoder\_dense(decoder\_outputs)*

*# Define model*

*model = Model([encoder\_inputs, decoder\_inputs], decoder\_outputs)*

*model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy')*

*# Prepare decoder target data*

*decoder\_target\_data = np.expand\_dims(fra\_padded, -1)*

*# Train model*

*model.fit([eng\_padded, fra\_padded], decoder\_target\_data, batch\_size=2, epochs=500, verbose=0)*

*# Simple prediction*

*test\_input = pad\_sequences(eng\_tokenizer.texts\_to\_sequences(['thank you']), maxlen=3, padding='post')*

*decoder\_input = np.zeros((1, 3)) # assume empty start for simplicity*

*preds = model.predict([test\_input, decoder\_input])*

*predicted\_ids = np.argmax(preds[0], axis=-1)*

*# Convert prediction to text*

*reverse\_fra\_word\_index = {i: word for word, i in fra\_tokenizer.word\_index.items()}*

*translated\_sentence = ' '.join([reverse\_fra\_word\_index.get(i, '') for i in predicted\_ids])*

*print("Translated Sentence:", translated\_sentence)*

**Output:**

****

**Conclusion:**

This program demonstrates the working of a basic Neural Machine Translation system using a sequence-to-sequence model. While the example is minimal for demonstration, scaling it up with larger datasets, implementing attention mechanisms, and applying post-processing can yield high-quality translations for real-world applications.