Translate English to Hindi translate Machine.

➢ Importing libraries:

The provided code snippet is used for importing necessary libraries and setting up the environment for a data processing and machine learning task.

- **NumPy and pandas:** These libraries are fundamental for data manipulation and analysis in Python. NumPy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Pandas, built on top of NumPy, offers data structures like DataFrames and Series, which are powerful tools for data manipulation and analysis.

- **os:** The os module provides a way to interact with the operating system, allowing the script to access files and directories. In this code, it is used to list files in the input directory.

- **string and re:** These libraries are used for text processing tasks. The string module provides a collection of string constants and functions for string manipulation. The re (regular expression) module is used for pattern matching with strings.

- **matplotlib.pyplot:** Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. pyplot is a sub-module that provides a MATLAB-like interface for creating plots and visualizations.

- **%matplotlib inline:** This is a magic command in Jupyter notebooks that allows plots to be displayed inline within the notebook.

- **seaborn:** Seaborn is a data visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

- **shuffle:** The shuffle function from sklearn.utils is used to shuffle the data. This is useful for randomizing the order of data points, which can be beneficial for model training to prevent any bias introduced by the order of the data.

- **train_test_split**: This function from sklearn.model_selection is used to split the dataset into training and testing sets. This is a common practice in machine learning to evaluate the performance of a model.

- **pd.set_option:** These lines set display options for pandas, such as the maximum number of rows and columns to display, the display width, and the maximum column width. This ensures that large datasets can be displayed properly in the output.

➢ Data pre-processing:

The provided code snippet is used for data preprocessing in a machine learning task, likely related to language translation. Here's a breakdown of the key steps:

Checking for Unique Values and Null Values:

- df['source'].value_counts(): Counts the occurrences of each unique value in the 'source' column.
- pd.isnull(df).sum(): Checks for null values in the DataFrame and sums them up.
- df=df[~pd.isnull(df['english_sentence'])]: Drops rows where the 'english_sentence' column is null.

Data Cleaning:

- df.drop_duplicates(inplace=True): Removes duplicate rows from the DataFrame.
- df=df.sample(n=25000,random_state=42): Randomly samples 25,000 rows from the DataFrame.
- Lowercasing: Converts all characters in the 'english_sentence' and 'hindi_sentence' columns to lowercase.
- Removing Quotes: Removes single quotes from the sentences.
- Removing Special Characters: Uses the string module to define a set of special characters to be removed from the sentences.
- Removing Numbers: Uses str.maketrans and translate to remove all numbers from the sentences.
- Removing Extra Spaces: Uses strip and re.sub to remove extra spaces from the sentences.
- Adding Start and End Tokens: Adds 'START_' at the beginning and '_END' at the end of each Hindi sentence.

Vocabulary Building:

- Creates sets of unique English and Hindi words from the sentences.
- len(english_words) and len(all_hindi_words): Returns the number of unique English and Hindi words, respectively.

Sentence Length Filtering:

Filters out sentences with lengths greater than 20 tokens in both English and Hindi.

Maximum Sentence Length Calculation:

Calculates the maximum length of sentences in both English and Hindi.

Token Indexing:

Builds token dictionaries for English and Hindi words.

Data Shuffling:

Shuffles the DataFrame to randomize the order of rows.

➢ Split The Data Into Train and Test:

Train-Test Split: Splits the data into training and testing sets using the train_test_split function from scikit-learn. It assigns 80% of the data to the training set (X_train, y_train) and 20% to the testing set (X_test, y_test).

Data Serialization: Saves the training and testing sets to pickle files (X_train.pkl, X_test.pkl) using the to_pickle method. This allows the data to be easily loaded and reused later.

Batch Data Generation Function (generate_batch):

- This function generates batches of data for training the model.
- It uses a generator to yield batches of data indefinitely.
- For each batch, it loops over the training data (X_train, y_train) in batch sizes (batch_size).
- It encodes the input and target sequences into one-hot encoded vectors (encoder_input_data, decoder_input_data, decoder_target_data).
- The input sequences are encoded using the input_token_index dictionary, and the target sequences are encoded using the target_token_index dictionary.
- The function returns a tuple containing the encoder and decoder input data arrays, along with the decoder target data array.

Overall, these operations prepare the data for training a sequence-to-sequence model, commonly used for tasks like machine translation.

The Encoder-Decoder Architecture[The Model]

Encoder: Uses an LSTM layer with a latent dimension of 300. It takes input sequences of English words and generates encoder states (hidden and cell states).

Decoder: Utilizes an LSTM layer with the same latent dimension. It takes the encoder states as initial states and processes the target Hindi sequences. The decoder outputs are passed through a dense softmax layer to generate the final predictions.

Model: The complete NMT model is defined using the encoder and decoder components, taking both English and Hindi sequences as inputs and producing Hindi sequences as outputs.

Training and Compilation:

- The model is compiled using the RMSprop optimizer and categorical crossentropy loss.
- It is trained using a custom data generator (generate_batch) to generate batches of training and validation data.

Inference and Evaluation:

- The trained model is used for inference to translate English sentences to Hindi.
- The decode_sequence function decodes input sequences using the trained model and prints the predicted Hindi translations along with the actual translations.

Summary

1. **Importing Libraries**: The code starts by importing necessary libraries such as NumPy, pandas, TensorFlow (via Keras), and others for data processing, visualization, and model building.

2. **Data Preprocessing:** The dataset is preprocessed to clean the text, remove special characters, and tokenize the sentences. Null values and duplicates are also handled. The English sentences are converted to lowercase, and start and end tokens are added to the Hindi sentences.

3. **Data Splitting:** The dataset is split into training and testing sets using the train_test_split function from scikit-learn.

4. **Model Architecture:** The NMT model architecture consists of an encoder-decoder architecture with LSTM layers. The encoder takes the input English sequences, embeds them, and produces encoder states. The decoder uses these states to generate the output Hindi sequences.

5. **Model Compilation:** The model is compiled with the RMSprop optimizer and categorical crossentropy loss, which is suitable for multi-class classification tasks like language translation.

6. **Training:** The model is trained using a custom data generator (generate_batch) to generate batches of training and validation data. The training loop runs for a specified number of epochs.

7. **Inference:** The trained model is used for inference. The decode_sequence function takes an input English sentence, encodes it using the encoder model, and then generates the corresponding Hindi translation using the decoder model.

8. **Model Evaluation:** Sample English sentences from the training set are decoded to check the model's performance. The input sentence, actual Hindi translation, and predicted Hindi translation are printed for comparison.

9. **Summary:** Overall, the code demonstrates the implementation of a basic NMT model for English to Hindi translation using LSTM layers in Keras. It covers data preprocessing, model building, training, and inference, providing a foundation for more advanced NMT models.