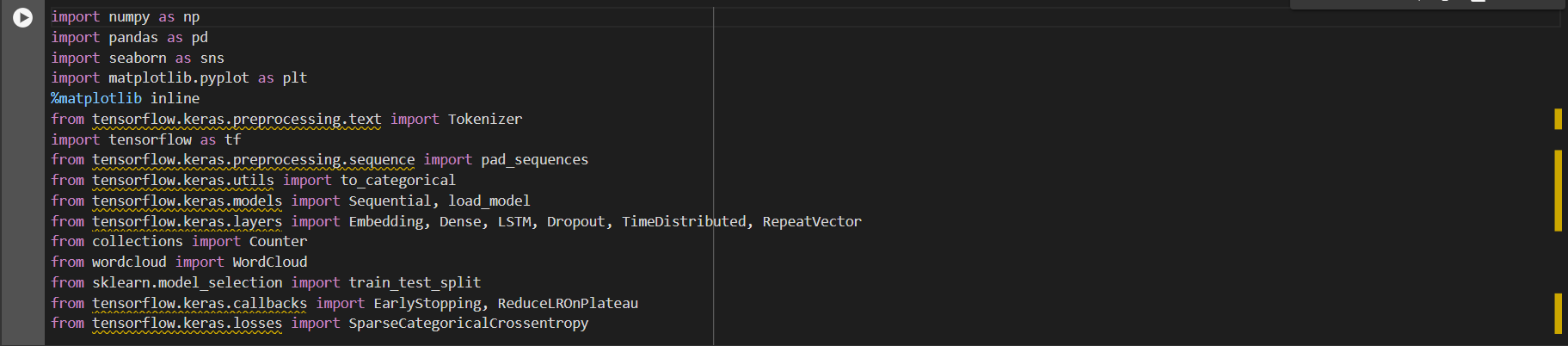
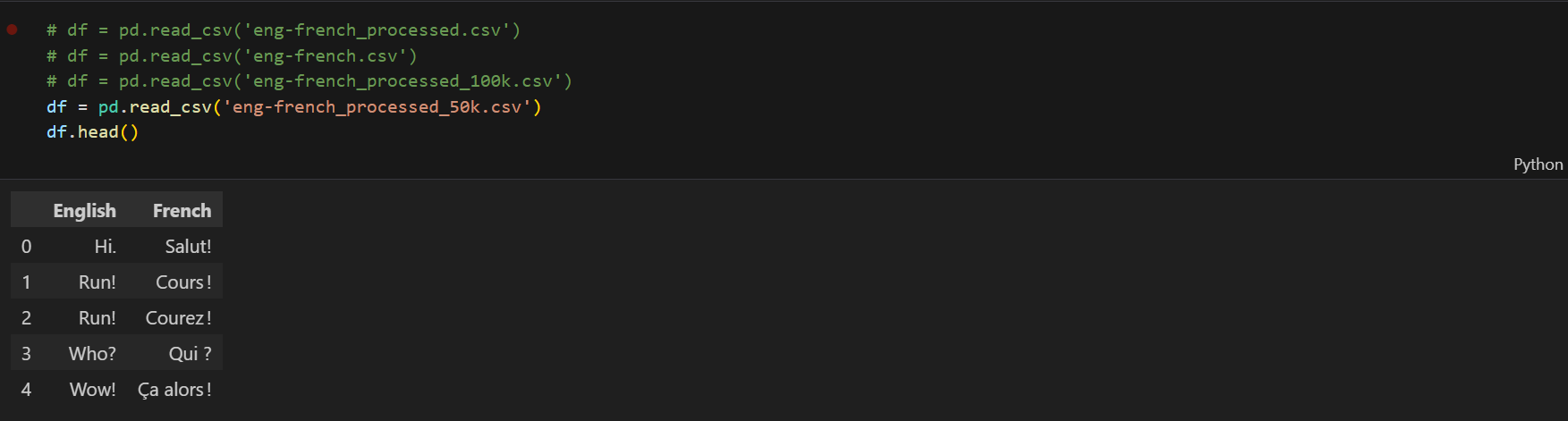
**Execution of English-to-French translation model**

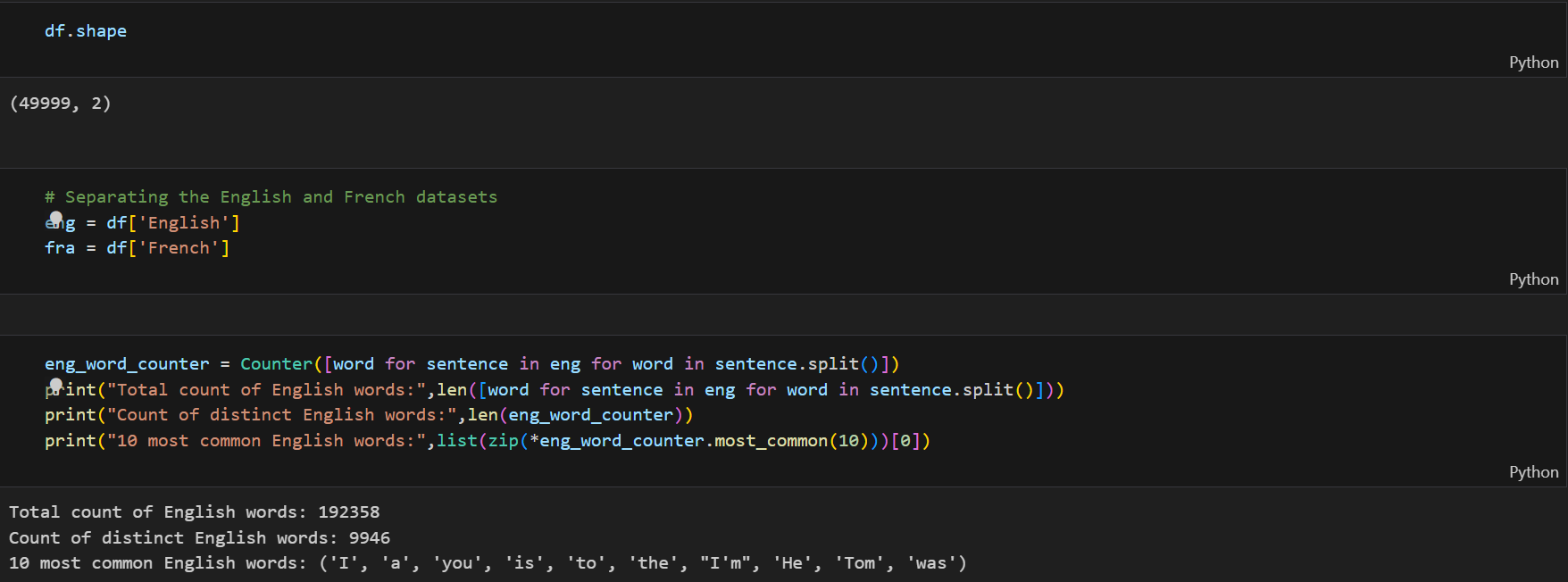


This section imports the core libraries and modules needed for building and training your English-to-French translation model. It includes tools for data manipulation, text preprocessing, neural network construction, and model evaluation:

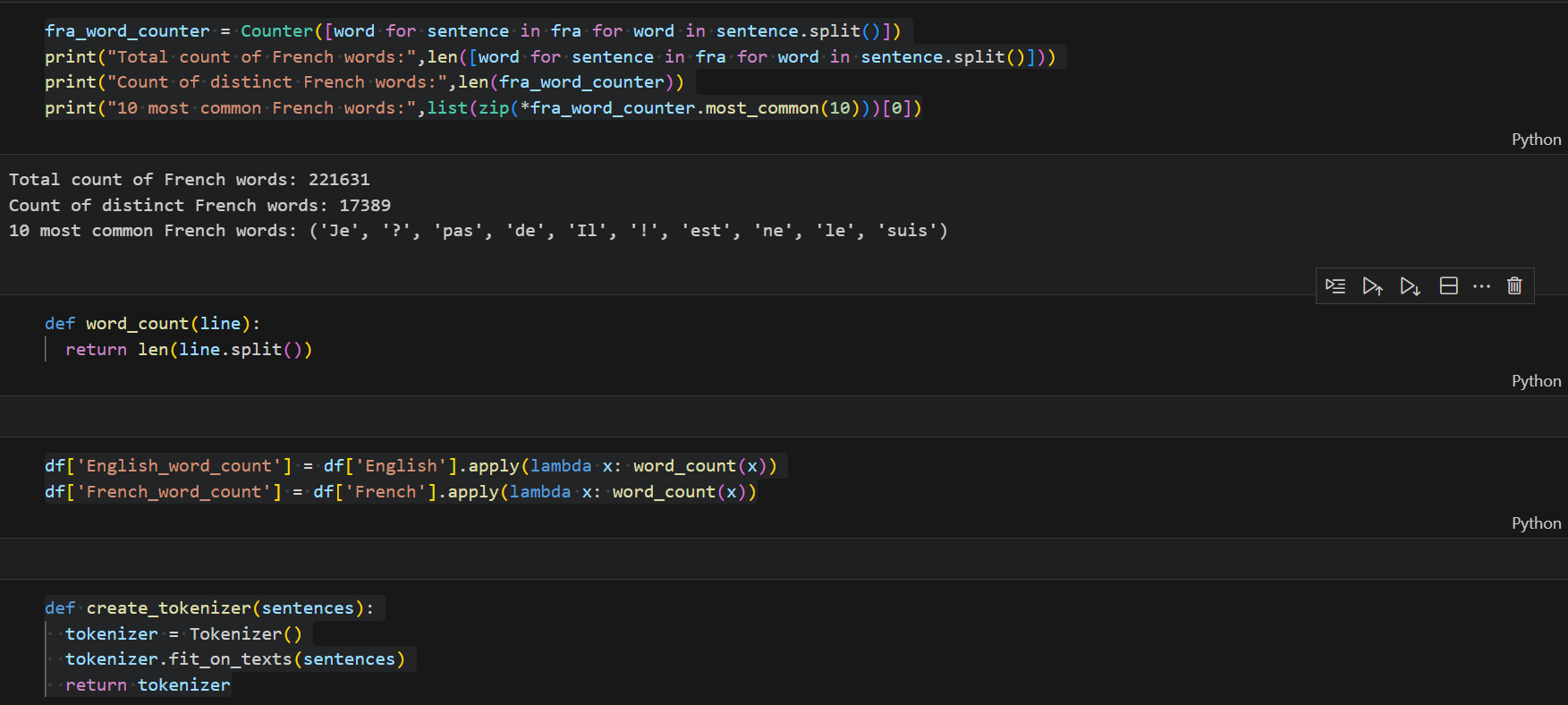
* **NumPy and Pandas**: For numerical operations and data handling.
* **Tokenizer**: To tokenize text data for neural network input.
* **TensorFlow and Keras**: The machine learning framework and its high-level API for building neural networks.
* **pad\_sequences**: Ensures uniform input length for neural network training.
* **Sequential, Embedding, Dense, LSTM, Dropout, TimeDistributed, RepeatVector**: Different types of neural network layers and models.
* **Counter**: For analyzing word frequency in the training data.
* **train\_test\_split**: Splits data into training and testing sets.
* **EarlyStopping, ReduceLROnPlateau**: Callbacks for controlling model training.
* **SparseCategoricalCrossentropy**: Loss function for multi-class classification.



This section loads an English-to-French translation dataset from a CSV file named 'eng-french\_processed\_50k.csv' using Pandas. It displays the first five rows of the dataset for initial inspection.



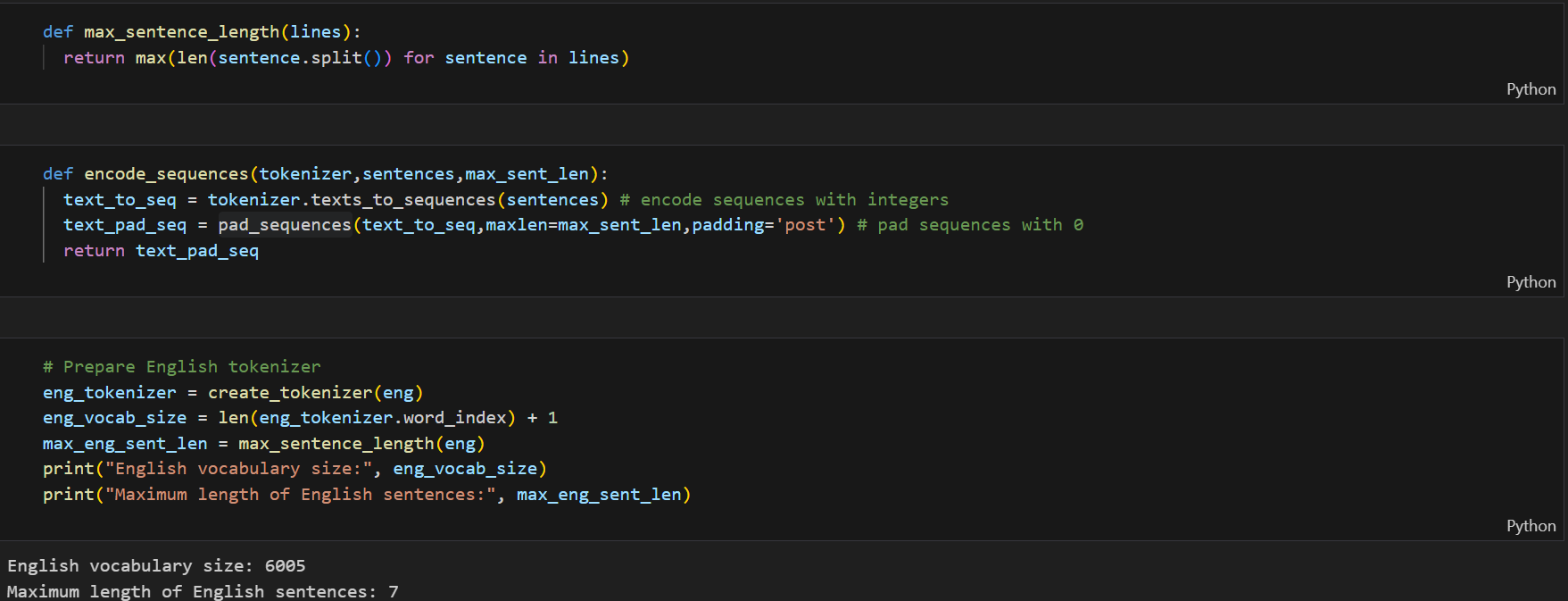
This section retrieves the shape of the dataset and separates the English and French datasets. It then calculates statistics about the English vocabulary, including the total count of words, the count of distinct words, and the 10 most common words.



This section calculates statistics for the French vocabulary, including total count, distinct count, and the 10 most common words.

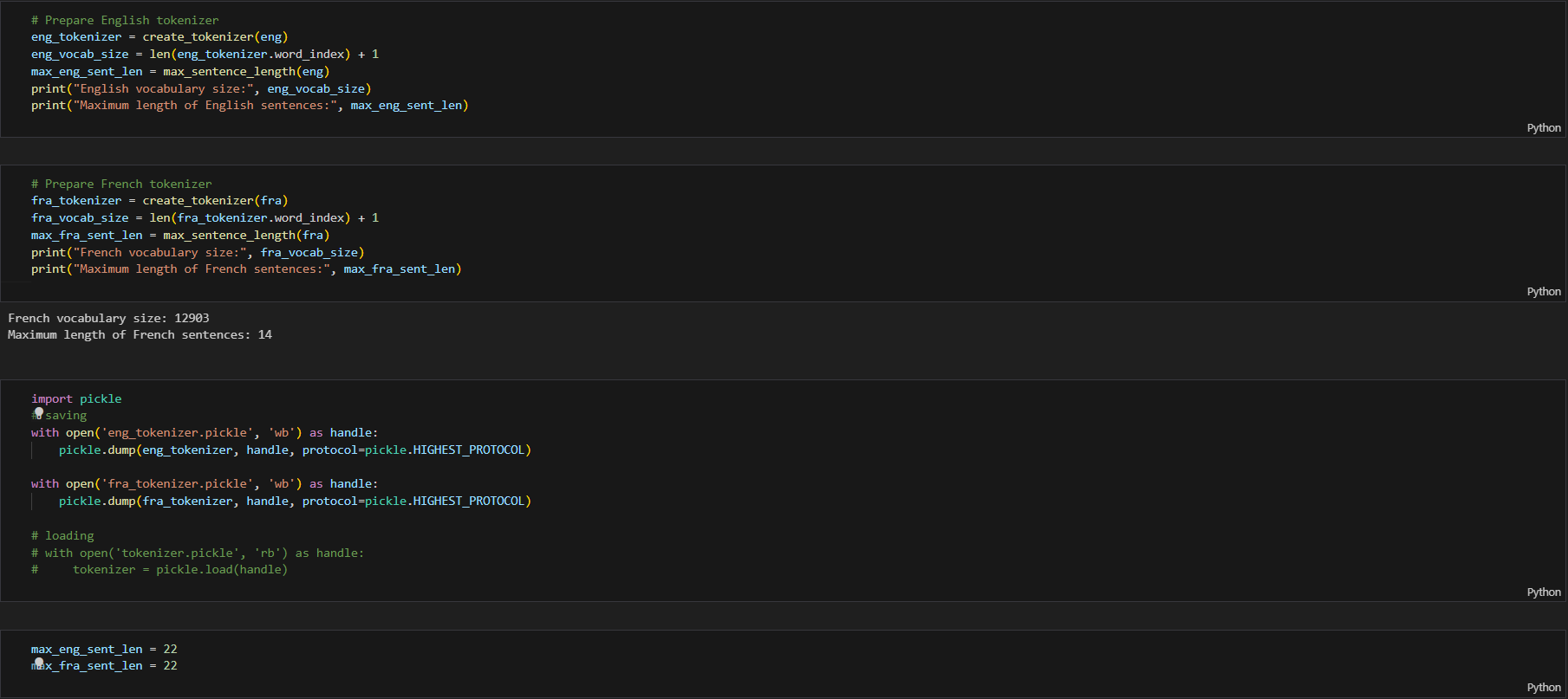
Two functions are defined:

1. **word\_count(line)**: Counts the number of words in a given line.
2. **create\_tokenizer(sentences)**: Creates a tokenizer object fitted to input sentences.



This section defines two functions:

1. **max\_sentence\_length(lines)**: This function calculates the maximum length of sentences in a given list of lines.
2. **encode\_sequences(tokenizer, sentences, max\_sent\_len)**: This function encodes input sentences into sequences of integers using a tokenizer, and then pads the sequences with zeros to ensure uniform length.

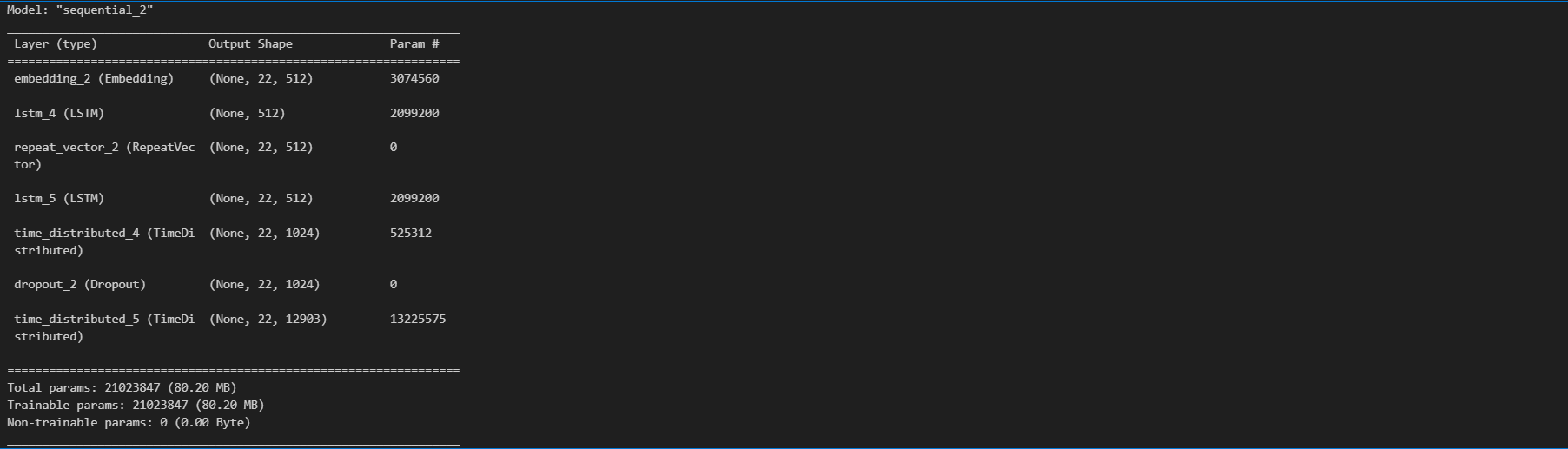
  
  
This section prepares tokenizers for both English and French sentences, calculates their respective vocabulary sizes and maximum sentence lengths, and then saves the tokenizers using pickle for future use.

The English tokenizer is created and its vocabulary size and maximum sentence length are determined. Similarly, the French tokenizer is created, and its vocabulary size and maximum sentence length are calculated.

Both tokenizers are then saved to files named 'eng\_tokenizer.pickle' and 'fra\_tokenizer.pickle' using pickle.

Finally, the maximum sentence lengths for both English and French are set to 22.

This section ensures that the tokenizers are ready for encoding input sentences into sequences of integers during the training of the translation model.

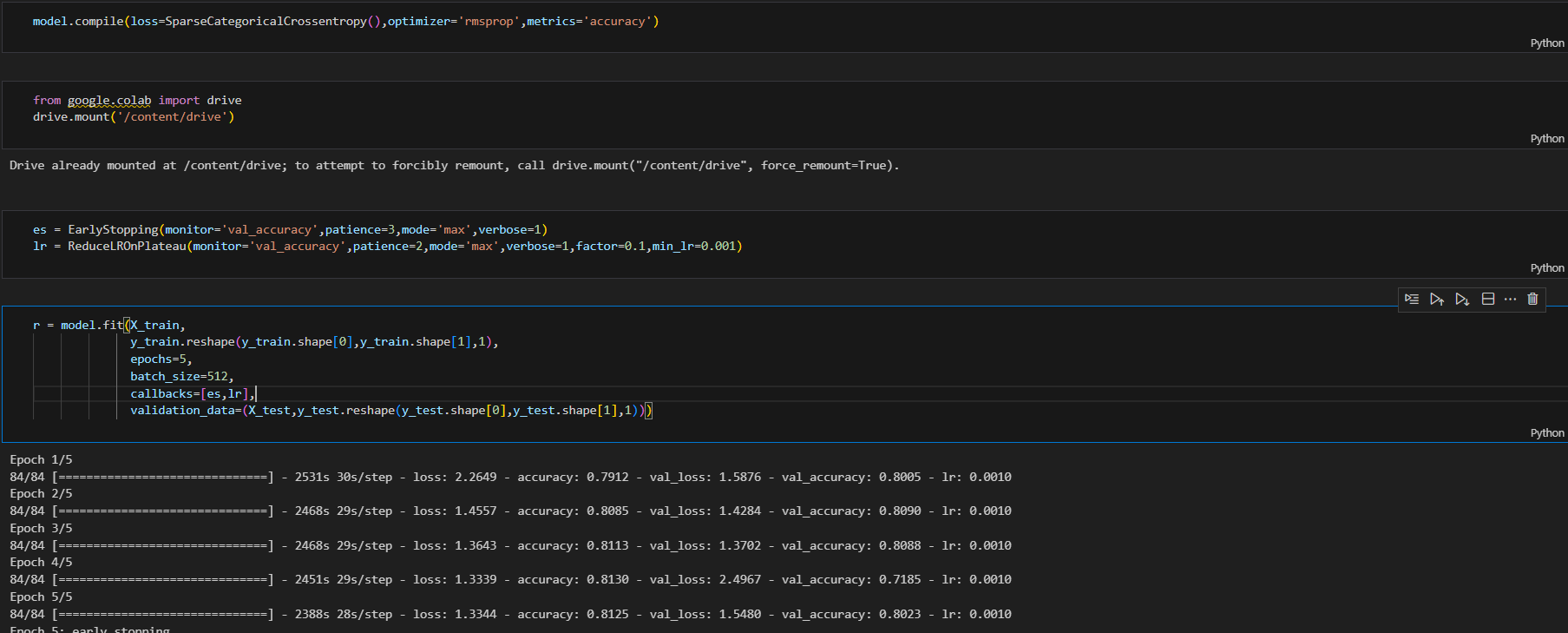
  
  
  
This section performs the encoding of sequences for both English and French sentences using the previously created tokenizers. It splits the data into training and testing sets using a 85%-15% ratio.

Next, a function **create\_model** is defined to construct the translation model. It is a sequential model with the following layers:

1. Embedding layer: Maps each word to a high-dimensional vector space.
2. LSTM layer: Processes the input sequence and captures contextual information.
3. RepeatVector layer: Repeats the output of the LSTM layer to match the desired output sequence length.
4. Another LSTM layer: Processes the repeated vectors to generate the output sequence.
5. TimeDistributed layer with Dense layer: Applies a dense layer to every temporal slice of the input.
6. Dropout layer: A regularization technique to prevent overfitting.
7. TimeDistributed layer with Dense layer: Applies a dense layer with softmax activation to predict the next word in each output sequence.

The model summary displays the architecture of the model, including the number of parameters in each layer.

This sets up the translation model for training on the encoded sequences of English and French sentences.



This section compiles the model using the SparseCategoricalCrossentropy loss function, RMSprop optimizer, and accuracy metric. It mounts Google Drive to save the model checkpoints during training.

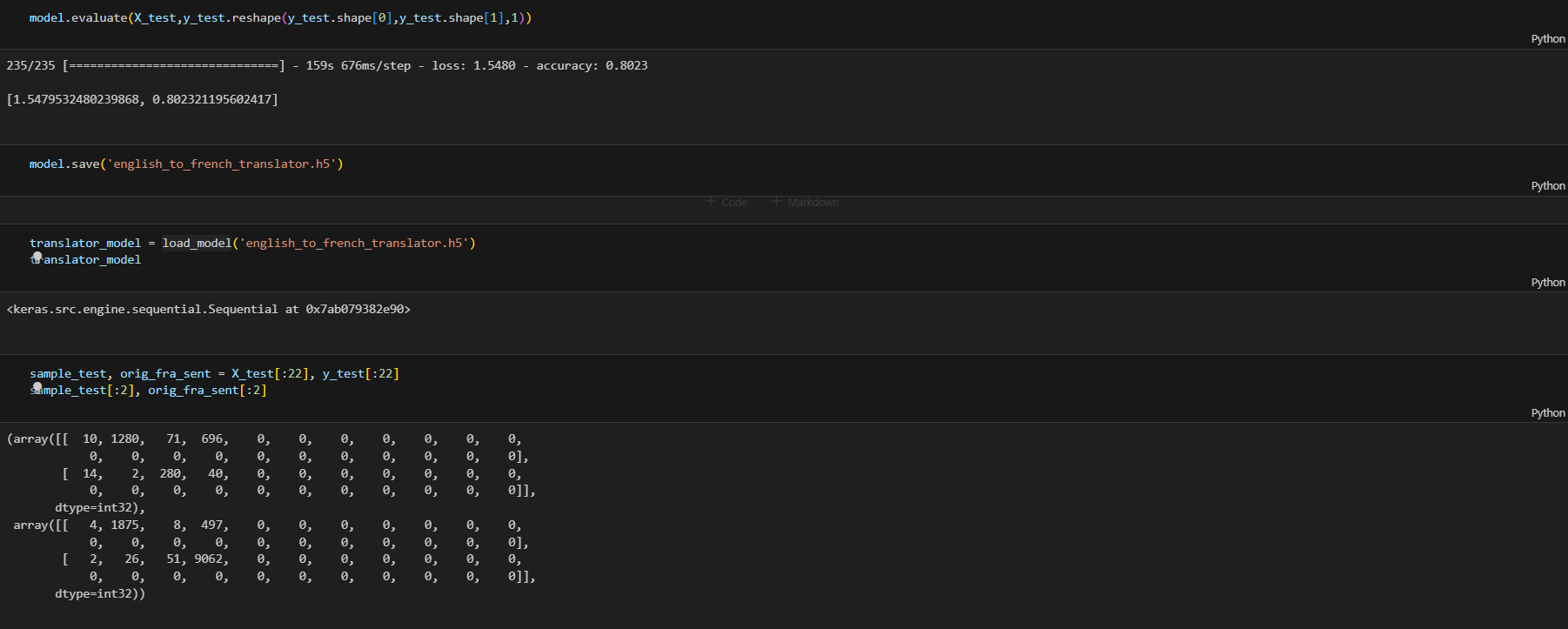
Two callbacks are defined:

1. EarlyStopping: Monitors validation accuracy and stops training if it doesn't improve for three consecutive epochs.
2. ReduceLROnPlateau: Monitors validation accuracy and reduces the learning rate by a factor of 0.1 if it doesn't improve for two consecutive epochs, with a minimum learning rate of 0.001.

The model is then trained using the fit() function:

* **X\_train** and **y\_train** are the training input and output sequences, respectively.
* The data is trained for 5 epochs with a batch size of 512.
* Callbacks are applied during training to implement early stopping and learning rate reduction.
* Validation data **(X\_test, y\_test)** is used to monitor the model's performance during training.

This section sets up and trains the translation model, monitoring its performance and adjusting the learning rate accordingly to improve accuracy.



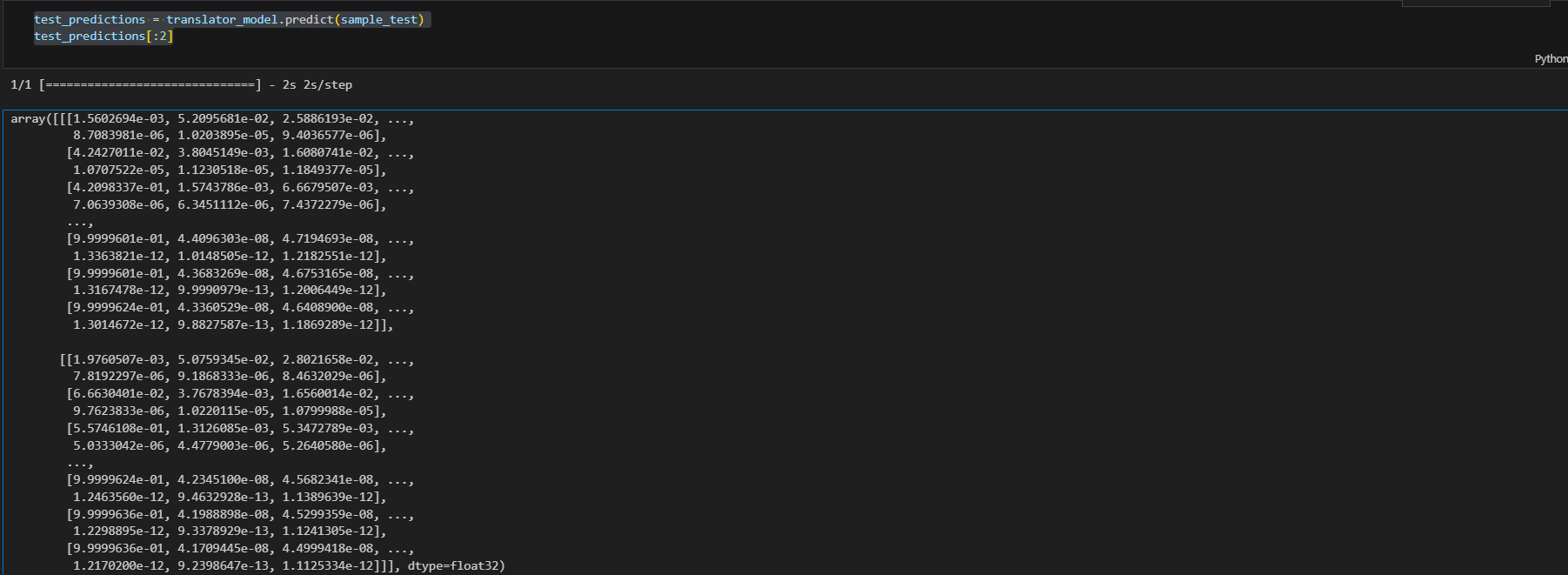
This section evaluates the trained model on the test data using the evaluate() function, which returns the loss value and accuracy metric.

The model is then saved to a file named 'english\_to\_french\_translator.h5' using the save() method.

After loading the saved model using load\_model(), the translator\_model variable holds the loaded model.

Lastly, a sample test input (sample\_test) and the corresponding original French sentences (orig\_fra\_sent) from the test data are extracted to demonstrate the model's translation performance.

This section completes the training and evaluation process of the translation model, and it prepares for the demonstration of translation on sample test data.



This section predicts translations for the sample test input (sample\_test) using the loaded translator\_model. The predictions are stored in the test\_predictions variable.

These predicted translations provide insight into how well the model performs on unseen data and serve as examples of the model's translation capability.



This section defines two helper functions:

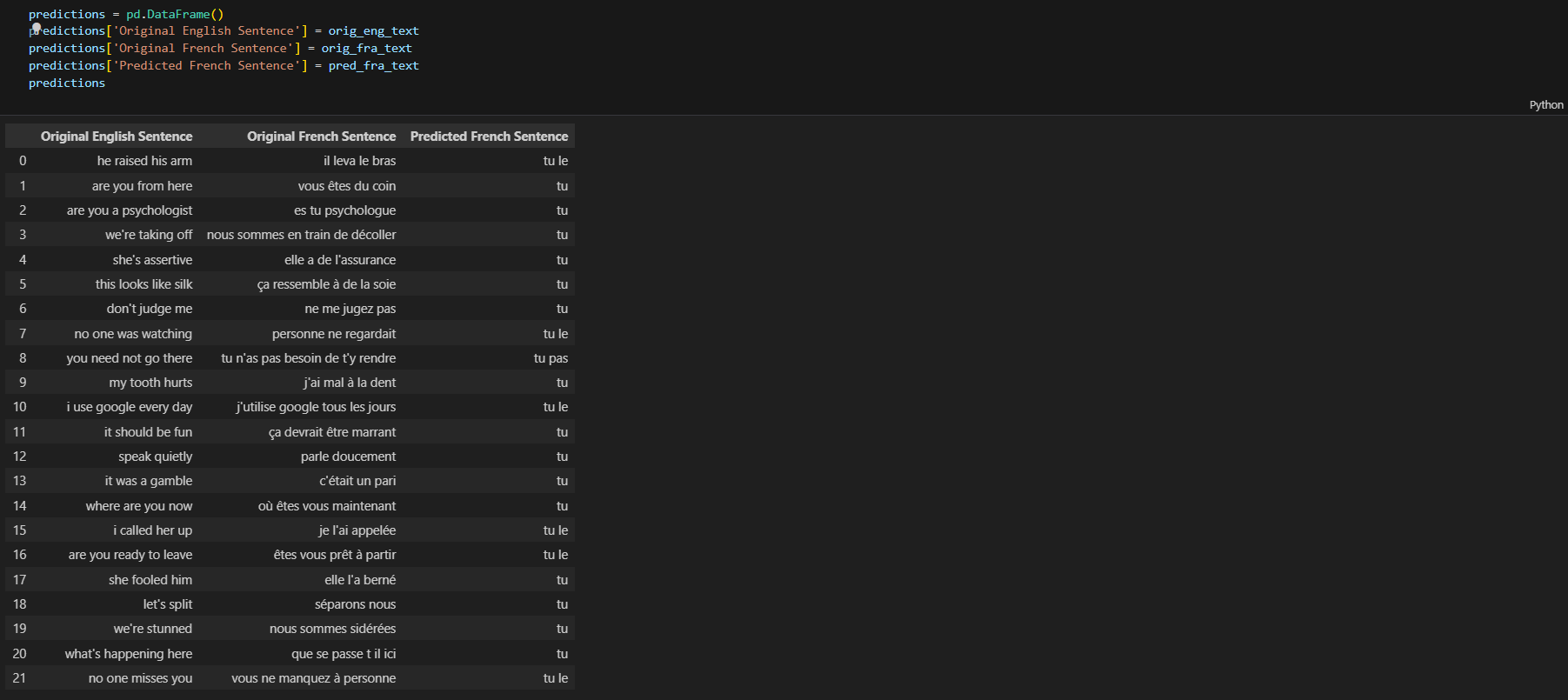
1. **convert\_pred\_to\_sent(input\_seq)**: Converts predicted indices to French sentences using the French tokenizer.
2. **convert\_idx\_to\_sent(input\_seq, tokenizer)**: Converts input indices to sentences using a given tokenizer.

It then initializes empty lists to store the original English text (**orig\_eng\_text**), original French text (**orig\_fra\_text**), and predicted French text (**pred\_fra\_text**).

A loop iterates over the sample test data to convert the input sequences, predicted sequences, and original French sequences into readable text format using the defined helper functions.

At the end of this section, the **orig\_eng\_text**, **pred\_fra\_text**, and **orig\_fra\_text** lists contain the corresponding English input, predicted French translations, and original French sentences, respectively.

These lists can be used to compare the model's translations with the ground truth and original text during the presentation.



This section creates a DataFrame named **predictions** containing three columns:

1. 'Original English Sentence': Contains the original English sentences.
2. 'Original French Sentence': Contains the original French sentences corresponding to the original English sentences.
3. 'Predicted French Sentence': Contains the predicted French translations generated by the model.

The DataFrame organizes the original and predicted sentences for easy comparison and analysis during the presentation.  
  
Excecution Instruction:  
To execute the model file on Google Colab, you can follow these steps:

1. **Upload the Model File**: First, upload the model file ('english\_to\_french\_translator.h5') to your Google Drive.
2. **Mount Google Drive**: Mount your Google Drive in the Colab notebook to access the uploaded model file. You can do this by executing the following code in a Colab cell:

pythonCopy code

from google.colab import drive drive.mount('/content/drive')

1. **Load the Model**: Load the model using the **load\_model** function from Keras. Ensure that you have imported the necessary libraries and modules, such as **load\_model**, **Tokenizer**, and any other dependencies required for model evaluation.

pythonCopy code

from tensorflow.keras.models import load\_model translator\_model = load\_model('/content/drive/My Drive/path/to/english\_to\_french\_translator.h5')

1. **Prepare Test Data**: Prepare your test data, if you haven't done so already. Ensure that it is in the appropriate format for input to the model.
2. **Make Predictions**: Use the loaded model to make predictions on the test data. Ensure that you preprocess the input data appropriately (e.g., tokenization, padding) before passing it to the model.

pythonCopy code

test\_predictions = translator\_model.predict(test\_data)

1. **Post-processing Predictions**: Post-process the predictions if necessary. For example, convert predicted indices to text using the tokenizer's index-word mapping.
2. **Evaluate Model Performance**: Evaluate the model's performance using appropriate metrics or compare the predictions with ground truth labels.
3. **Visualize Results**: Visualize the results, such as comparing original and predicted translations, to assess the model's accuracy and effectiveness.