English to French Translation

STEP 1: IMPORTING LIBRARIES

```
!pip install tensorflow
!pip install --upgrade tensorflow-gpu==2.0
!pip install nltk
!pip install gensim

!pip install spacy
!pip install plotly
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install seaborn
!pip install wordcloud
!pip install jupyterthemes
!pip install sklearn

import pandas as pd
import numpy as np
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/pyt
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.3
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensor
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorfl
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.1
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorb
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorbo
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorb
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-au
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth
ERROR: Could not find a version that satisfies the requirement tensorflow-gpu==2.0 (from versions: 2.8.0rc0, 2.8.0rc1, 2.8.0, 2.8.1,
ERROR: No matching distribution found for tensorflow-gpu==2.0
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.4)
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.25.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from gensim) (1.11.4)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim) (6.4.0)
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (3.7.4)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.10)
```

```python
from collections import Counter
import operator
import plotly.express as px
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
import nltk
import re
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from tensorflow.keras.preprocessing.text import one_hot, Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, TimeDistributed, RepeatVector, Embedding, Input, LSTM, Conv1D, MaxPool1D, Bidirectional
from tensorflow.keras.models import Model
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
```

STEP 2: LOADING THE DATASET

```python
df_english = pd.read_csv('vocab_en.csv', sep = '\t', names = ['english'])
df_french = pd.read_csv('vocab_fr.csv', sep = '\t', names = ['french'])
```

```python
# CHECKING IF NULL ELEMENTS ARE PRESENT OR NOT IN BOTH DATASETS
df_english.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137860 entries, 0 to 137859
Data columns (total 1 columns):
 #   Column   Non-Null Count   Dtype 
---  ------   --------------   ----- 
 0   english  137860 non-null  object
dtypes: object(1)
memory usage: 1.1+ MB
```

```python
df_french.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137860 entries, 0 to 137859
Data columns (total 1 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   french  137860 non-null  object
dtypes: object(1)
memory usage: 1.1+ MB
```

```python
# THUS WE HAVE NO NULL ELEMENTS PRESENT
# CONCATENATING BOTH THE DATAFRAMES (df_english and df_french)
df = pd.DataFrame([df_english['english'],df_french['french']])
df = df.T
df
```

	english	french
0	new jersey is sometimes quiet during autumn , ...	new jersey est parfois calme pendant l' automn...
1	the united states is usually chilly during jul...	les états-unis est généralement froid en juill...
2	california is usually quiet during march , and...	california est généralement calme en mars , et...
3	the united states is sometimes mild during jun...	les états-unis est parfois légère en juin , et...
4	your least liked fruit is the grape , but my l...	votre moins aimé fruit est le raisin , mais mo...
...	...	...
137855	france is never busy during march , and it is ...	la france est jamais occupée en mars , et il e...
137856	india is sometimes beautiful during spring , a...	l' inde est parfois belle au printemps , et il...
137857	india is never wet during summer , but it is	l' inde est jamais mouillé pendant l' été , ma

STEP 3: PERFORMING DATA CLEANING

```
# REMOVING PUNCTUATIONS FROM OUR TEXT

def remove_punc(x):
    return re.sub('[!#?,.:";"]', "", x)

df['french'] = df['french'].apply(remove_punc)
df['english'] = df['english'].apply(remove_punc)


# CHECKINH HOW MANY UNIQUE WORDS ARE PRESENT IN THE ENGLISH DICTIONARY
english_words = []
for i in df['english'].values:
    for j in i.split(" "):
        english_words.append(j)
english_words = list(set(english_words))
english_words = english_words[1:]

len(english_words)
```

```
199
```

```
french_words = []
for i in df['french'].values:
    for j in i.split(" "):
        french_words.append(j)
french_words = list(set(french_words))
french_words = french_words[1:]

len(french_words)
```

```
350
```

```
english_words
```

```
['sometimes',
 'spanish',
 'november',
 'nice',
 'california',
 'driving',
 'busy',
 'mangoes',
 "aren't",
 'mouse',
 'favorite',
 'want',
 'her',
 'we',
 'like',
 'mild',
 'plan',
 'do',
 'march',
 'june',
 'grapes',
 'wanted',
 'animal',
 'elephant',
 'are',
 'translate',
 'peaches',
 'football',
 "it's",
 'autumn',
 'monkeys',
 'went',
 'she',
 'where',
 'tower',
 'sharks',
 'birds',
 'cold',
 'lime',
 'feared',
 'winter',
 'dog',
 'plans',
 'united',
 'portuguese',
 'mango',
 'difficult',
 'that',
 'dogs',
 'rusty',
 'grocery',
 'lake',
 'september',
 'chilly',
 'fun',
 'it',
 'strawberries',
 'thinks',
```

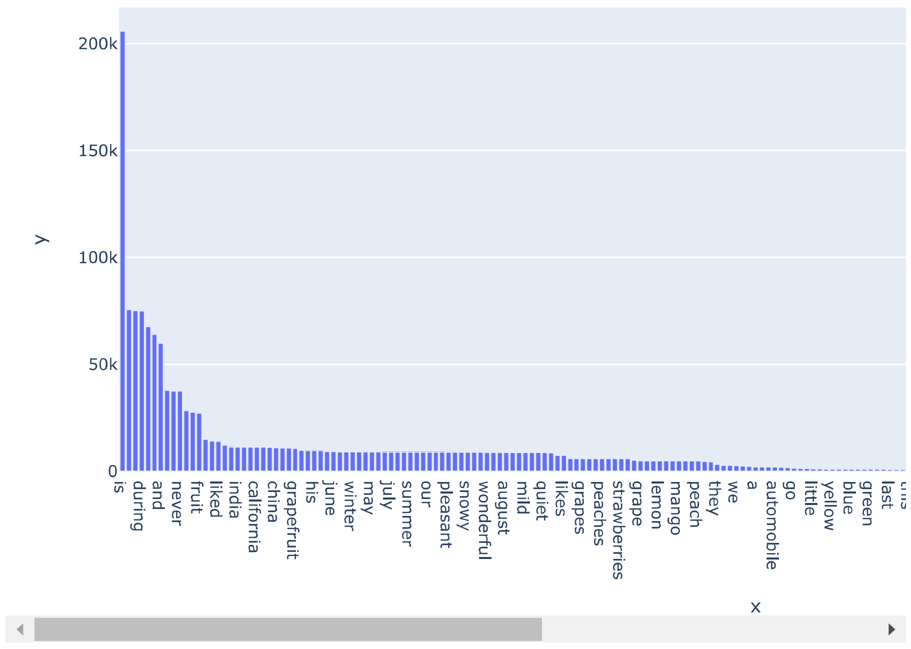
STEP 4: VISUALIZING DATASET

```
# GETING FREQUENCY OF EACH WORD
words = []
for i in df['english']:
    for word in i.split():
        words.append(word)

english_words_counts = Counter(words)

english_words_counts = sorted(english_words_counts.items(), key = operator.itemgetter(1), reverse = True)
```

```
english_words_counts
```

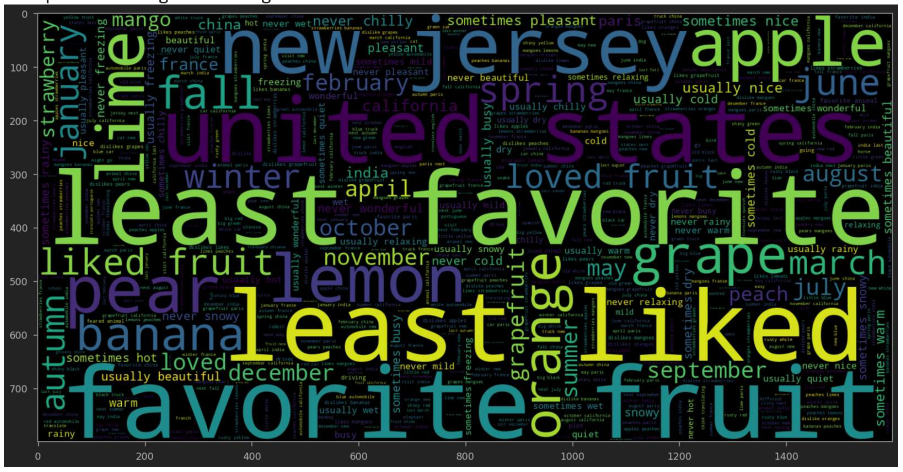
```
[('is', 205858),
 ('in', 75525),
 ('it', 75137),
      ('during', 74933),
      ('the', 67628),
      ('but', 63987),
      ('and', 59850),
      ('sometimes', 37746),
      ('usually', 37507),
      ('never', 37500),
      ('favorite', 28332),
      ('least', 27564),
      ('fruit', 27192),
      ('most', 14934),
      ('loved', 14166),
      ('liked', 14046),
      ('new', 12197),
      ('paris', 11334),
      ('india', 11277),
      ('united', 11270),
      ('states', 11270),
      ('california', 11250),
      ('jersey', 11225),
      ('france', 11170),
      ('china', 10953),
      ('he', 10786),
      ('she', 10786),
      ('grapefruit', 10692),
      ('your', 9734),
      ('my', 9700),
      ('his', 9700),
      ('her', 9700),
      ('fall', 9134),
      ('june', 9133),
      ('spring', 9102),
      ('january', 9090),
      ('winter', 9038),
      ('march', 9023),
      ('autumn', 9004),
      ('may', 8995),
      ('nice', 8984),
      ('september', 8958),
      ('july', 8956),
      ('april', 8954),
      ('november', 8951),
      ('summer', 8948),
      ('december', 8945),
      ('february', 8942),
      ('our', 8932),
      ('their', 8932),
      ('freezing', 8928),
      ('pleasant', 8916),
      ('beautiful', 8915),
      ('october', 8910),
      ('snowy', 8898),
      ('warm', 8890),
      ('cold', 8878),
      ('wonderful', 8808),
```

```
# APPENDING VALUES TO DIFFERENT LISTS FOR VISUALIZATION PURPOSES
english_words = []
english_counts = []
for i in range(len(english_words_counts)):
    english_words.append(english_words_counts[i][0])
    english_counts.append(english_words_counts[i][1])


# PLOTTING BARPLOT USING PLOTLY
fig = px.bar(x = english_words, y = english_counts)
fig.show()
```



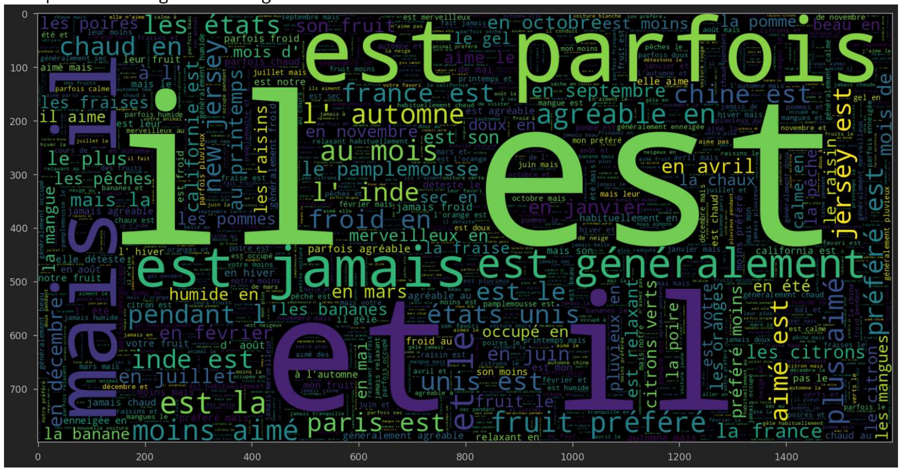
```
# PLOTTING A WORDCLOUD FOR ENGLISH
plt.figure(figsize = (20, 20))
wc = WordCloud(max_words = 2000, width = 1600, height = 800).generate(" ".join(df.english))


plt.imshow(wc, interpolation='bilinear')
```

```
<matplotlib.image.AxesImage at 0x7f563d30ab90>
```



```
# PLOTTING WORDCLOUD FOR FRENCH LANGUAGE
plt.figure(figsize = (20, 20))
wc = WordCloud(max_words = 2000, width = 1600, height = 800).generate(" ".join(df.french))


plt.imshow(wc, interpolation='bilinear')
```

```
<matplotlib.image.AxesImage at 0x7f563d119090>
```



```
# FINDING THE MAXIMUM LENGTH IN THE ENGLISH DATAFRAME

maxlen_english = 0
for doc in df.english:
    tokens = nltk.word_tokenize(doc)
    if maxlen_english < len(tokens):
        maxlen_english = len(tokens)
maxlen_english
```

```
15
```

```
# FINDING THE MAXIMUM LENGTH IN THE FRENCH DATAFRAME

maxlen_french = 0
for doc in df.french:
    tokens = nltk.word_tokenize(doc)
    if maxlen_french< len(tokens):
        maxlen_french = len(tokens)
maxlen_french
```

```
23
```

STEP 5: PREPARING THE DATA BY PERFORMING TOKENIZING AND PADDING

```
# CONVERTING OUR TEXT TO NUMBERS TO BUILD AI MODEL

def tokenize_and_pad(x, maxlen):
    tokenizer = Tokenizer(char_level = False)
    tokenizer.fit_on_texts(x)
    sequences = tokenizer.texts_to_sequences(x)
    padded = pad_sequences(sequences, maxlen = maxlen, padding = 'post')  #TO MAKE LENGTH OF EACH TOKENIZED TEXT EQUAL
    return tokenizer, sequences, padded

x_tokenizer, x_sequences, x_padded = tokenize_and_pad(df.english, maxlen_english)
y_tokenizer, y_sequences, y_padded = tokenize_and_pad(df.french, maxlen_french)
```

```
# TRAIN AND TEST SPLITTING THE DATASET
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_padded, y_padded, test_size = 0.1)
```

STEP 6: BUILDING AND TRAINING THE LSTM MODEL

```
# TOTAL VOCAB SIZE, SINCE WE ADDED PADDING QE ADD 1 TO THE TOTAL WORD COUNT

english_vocab_size = len(english_words) + 1
french_vocab_size = len(french_words) + 1

# SEQUENTIAL MODEL
model = Sequential()

# ADDING EMBEDDING LAYER

model.add(Embedding(english_vocab_size, 256, input_length = maxlen_english, mask_zero = True))
model.add(LSTM(256))

# DECODER
# ADDING REPEAT VECTOR

model.add(RepeatVector(maxlen_french))
model.add(LSTM(256, return_sequences = True))
model.add(TimeDistributed(Dense(french_vocab_size, activation = 'softmax')))
model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 15, 256)           51200

 lstm (LSTM)                 (None, 256)               525312

 repeat_vector (RepeatVecto  (None, 23, 256)           0
 r)

 lstm_1 (LSTM)               (None, 23, 256)           525312

 time_distributed (TimeDist  (None, 23, 351)           90207
 ributed)

=================================================================
Total params: 1192031 (4.55 MB)
Trainable params: 1192031 (4.55 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# CHANGING SHAPE OF TARGET FROM 2D TO 3D
y_train = np.expand_dims(y_train, axis = 2)
y_train.shape
```

```
(124074, 23, 1)
```

```
# FINALLY TRAINING THE MODEL

model.fit(X_train, y_train, batch_size = 1024, validation_split = 0.1, epochs = 15)
```

```
Epoch 1/15
110/110 [==============================] - 376s 3s/step - loss: 2.6974 - accuracy: 0.4968 - val_loss: 2.1300 - val_accuracy: 0.5387
Epoch 2/15
110/110 [==============================] - 346s 3s/step - loss: 1.8797 - accuracy: 0.5670 - val_loss: 1.6485 - val_accuracy: 0.5959
Epoch 3/15
110/110 [==============================] - 352s 3s/step - loss: 1.5167 - accuracy: 0.6162 - val_loss: 1.4189 - val_accuracy: 0.6232
Epoch 4/15
110/110 [==============================] - 357s 3s/step - loss: 1.2994 - accuracy: 0.6539 - val_loss: 1.2311 - val_accuracy: 0.6587
Epoch 5/15
110/110 [==============================] - 350s 3s/step - loss: 1.1356 - accuracy: 0.6813 - val_loss: 1.0615 - val_accuracy: 0.6987
Epoch 6/15
110/110 [==============================] - 352s 3s/step - loss: 0.9906 - accuracy: 0.7168 - val_loss: 0.9273 - val_accuracy: 0.7346
Epoch 7/15
110/110 [==============================] - 352s 3s/step - loss: 0.8779 - accuracy: 0.7480 - val_loss: 0.9047 - val_accuracy: 0.7352
Epoch 8/15
110/110 [==============================] - 356s 3s/step - loss: 0.7879 - accuracy: 0.7737 - val_loss: 0.7410 - val_accuracy: 0.7881
Epoch 9/15
110/110 [==============================] - 353s 3s/step - loss: 0.7001 - accuracy: 0.7987 - val_loss: 0.6710 - val_accuracy: 0.8059
Epoch 10/15
110/110 [==============================] - 350s 3s/step - loss: 0.6311 - accuracy: 0.8193 - val_loss: 0.6136 - val_accuracy: 0.8210
```

```
Epoch 11/15
110/110 [==============================] - 345s 3s/step - loss: 0.5685 - accuracy: 0.8375 - val_loss: 0.5475 - val_accuracy: 0.8413
Epoch 12/15
110/110 [==============================] - 360s 3s/step - loss: 0.5164 - accuracy: 0.8516 - val_loss: 0.5358 - val_accuracy: 0.8423
Epoch 13/15
110/110 [==============================] - 364s 3s/step - loss: 0.4629 - accuracy: 0.8667 - val_loss: 0.4424 - val_accuracy: 0.8719
Epoch 14/15
110/110 [==============================] - 367s 3s/step - loss: 0.4126 - accuracy: 0.8812 - val_loss: 0.4183 - val_accuracy: 0.8786
Epoch 15/15
110/110 [==============================] - 349s 3s/step - loss: 0.3677 - accuracy: 0.8946 - val_loss: 0.4026 - val_accuracy: 0.8798
<keras.src.callbacks.History at 0x7f56279c85e0>
```

```
# SAVING THE MODEL

model.save("Translator.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning:

You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the nat
```

STEP 7: ASSES TRAINED MODEL PERFORMANCE

```
y_predict = model.predict(X_test)

def prediction(X, x_tokenizer=x_tokenizer, y_tokenizer=y_tokenizer):
    predictions = model.predict(X)
    id_to_word = {id: word for word, id in y_tokenizer.word_index.items()}
    id_to_word[0] = ''
    predicted_indices = np.argmax(predictions, axis=2).flatten()
    return ' '.join([id_to_word[j] for j in predicted_indices if j != 0])

# Define the function to convert padded sequences back to text
def pad_to_text(padded, tokenizer):
    id_to_word = {id: word for word, id in tokenizer.word_index.items()}
    id_to_word[0] = ''
    return ' '.join([id_to_word[j] for j in padded if j != 0])

# Print original and predicted texts for the first 5 samples
for i in range(5):
    print('Original English Text - {}\n'.format(pad_to_text(X_test[i], x_tokenizer)))
    print('Original French Text - {}\n'.format(pad_to_text(y_test[i], y_tokenizer)))
    print('Predicted French Text - {}\n\n\n'.format(prediction(X_test[i:i+1])))
```

```
431/431 [==============================] - 25s 59ms/step
Original English Text - california is never busy during october and it is never snowy in march

Original French Text - california est jamais occupé en octobre et il est jamais de neige en mars

1/1 [==============================] - 0s 34ms/step
Predicted French Text - california est jamais occupé en octobre et il est jamais en en mars



Original English Text - france is usually freezing during february but it is usually wonderful in october

Original French Text - la france est le gel habituellement en février mais il est généralement merveilleux en octobre

1/1 [==============================] - 0s 39ms/step
Predicted French Text - la france est le gel habituellement en février il est généralement généralement en octobre



Original English Text - india is usually beautiful during august and it is never snowy in september

Original French Text - l' inde est généralement beau au mois d' août et il est jamais neigeux en septembre

1/1 [==============================] - 0s 33ms/step
Predicted French Text - l' inde est généralement calme au mois d' août et il est jamais en septembre



Original English Text - paris is usually chilly during january but it is never busy in spring

Original French Text - paris est généralement froid en janvier mais il est jamais occupé au printemps

1/1 [==============================] - 0s 37ms/step
```

```
Predicted French Text - paris est généralement froid en janvier mais il est jamais froid au printemps


Original English Text - the united states is freezing during august but it is hot in october

Original French Text - les états unis est le gel au mois d' août mais il est chaud en octobre
```