In [1]:
```
import string
import re
import pandas as pd
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, RepeatVector
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from keras import optimizers
```

In [2]:
```
data_path = 'C:/Users/91947/OneDrive/Desktop/fra-eng/fra.txt'
with open(data_path, 'r', encoding='utf-8') as f:
    lines = f.read()
```

In [4]:
```
#splitting into lines and words for preprocessing
def to_lines(text):
    sents = text.strip().split('\n')
    sents = [i.split('\t') for i in sents]
    return sents
```

In [ ]:
```
lines
```

In [5]:
```
fra_eng = to_lines(lines)
fra_eng[:5]
```

Out[5]:
```
[['Go.', 'Va !'],
 ['Hi.', 'Salut !'],
 ['Run!', 'Cours\u202f!'],
 ['Run!', 'Courez\u202f!'],
 ['Who?', 'Qui ?']]
```

In [6]:
```
#Converting into array
import numpy as np

fra_eng = np.array(fra_eng)
fra_eng[:5]
```

Out[6]:
```
array([['Go.', 'Va !'],
       ['Hi.', 'Salut !'],
       ['Run!', 'Cours\u202f!'],
       ['Run!', 'Courez\u202f!'],
       ['Who?', 'Qui ?']], dtype='<U349')
```

In [7]:
```
#Selecting only 50000 records for fast processing as the data set is too large
fra_eng = to_lines(lines)
fra_eng = np.array(fra_eng)[:50000, [0, 1]]
```

In [8]:
```
#DATACLEANING
#remove punctuation
fra_eng[:, 0] = [s.translate(str.maketrans('', '', string.punctuation)).lower() for
fra_eng[:, 1] = [s.translate(str.maketrans('', '', string.punctuation)).lower() for
```

In [9]:
```
#TEXT TO SEQUENCE CONVERSION (WORD TO INDEX MAPPING)

#function to build a tokenizer
# Tokenization
def tokenization(lines):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
```

```
    return tokenizer

eng_tokenizer = tokenization(fra_eng[:, 0])
eng_vocab_size = len(eng_tokenizer.word_index) + 1
eng_length = 8

fra_tokenizer = tokenization(fra_eng[:, 1])
fra_vocab_size = len(fra_tokenizer.word_index) + 1
fra_length = 8
```

In [10]:
```
# Data encoding
def encode_sequences(tokenizer, length, lines):
    seq = tokenizer.texts_to_sequences(lines)
    seq = pad_sequences(seq, maxlen=length, padding='post')
    return seq
```

In [11]:
```
# Split data into train and test sets
train, test = train_test_split(fra_eng, test_size=0.2, random_state=12)
trainX = encode_sequences(fra_tokenizer, fra_length, train[:, 1])
trainY = encode_sequences(eng_tokenizer, eng_length, train[:, 0])
testX = encode_sequences(fra_tokenizer, fra_length, test[:, 1])
testY = encode_sequences(eng_tokenizer, eng_length, test[:, 0])
```

In [12]:
```
# Define the NMT model
def define_model(input_vocab, output_vocab, input_timesteps, output_timesteps, unit
    model = Sequential()
    model.add(Embedding(input_vocab, units, input_length=input_timesteps, mask_zero
    model.add(LSTM(units))
    model.add(RepeatVector(output_timesteps))
    model.add(LSTM(units, return_sequences=True))
    model.add(Dense(output_vocab, activation='softmax'))
    return model
```

In [13]:
```
#creating an encoder-decoder architecture for neural machine translation.
model = define_model(fra_vocab_size, eng_vocab_size, fra_length, eng_length, units=
```

In [14]:
```
# Compile the model
optimizer = optimizers.RMSprop(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy')
```

In [15]:
```
# Train the model
model.fit(trainX, trainY, epochs=10, batch_size=512, validation_split=0.2)
```

```
Epoch 1/10
63/63 [==============================] - 177s 3s/step - loss: 4.4844 - val_loss:
3.3870
Epoch 2/10
63/63 [==============================] - 163s 3s/step - loss: 3.1783 - val_loss:
3.0796
Epoch 3/10
63/63 [==============================] - 165s 3s/step - loss: 3.0180 - val_loss:
2.9943
Epoch 4/10
63/63 [==============================] - 162s 3s/step - loss: 2.9661 - val_loss:
2.9873
Epoch 5/10
63/63 [==============================] - 12174s 196s/step - loss: 2.9408 - val_los
s: 2.9611
Epoch 6/10
63/63 [==============================] - 157s 2s/step - loss: 2.9211 - val_loss:
2.9299
Epoch 7/10
63/63 [==============================] - 160s 3s/step - loss: 2.9047 - val_loss:
2.9179
Epoch 8/10
63/63 [==============================] - 160s 3s/step - loss: 2.8905 - val_loss:
2.9248
Epoch 9/10
63/63 [==============================] - 161s 3s/step - loss: 2.8777 - val_loss:
2.8974
Epoch 10/10
63/63 [==============================] - 163s 3s/step - loss: 2.8673 - val_loss:
2.8984
```

Out[15]:
```
<keras.src.callbacks.History at 0x219060890d0>
```

In [16]:
```
preds = model.predict(testX)
```

```
313/313 [==============================] - 43s 123ms/step
```

In [17]:
```
#these predictions are sequences of integers. We need to convert these integers to
def get_word(n, tokenizer):
    return tokenizer.index_word.get(n)
```

In [18]:
```
#convert predictions into sentences(English)
max_length = eng_length
preds_text = []

for i in preds:
    temp = []
    for j in range(max_length):
        if j < len(i):
            t = get_word(np.argmax(i[j]), eng_tokenizer)
            if j > 0:
                if (t == get_word(np.argmax(i[j - 1]), eng_tokenizer)) or (t is Nor
                    temp.append(' ')
                else:
                    temp.append(t)
            else:
                if t is None:
                    temp.append(' ')
                else:
                    temp.append(t)
        else:
            temp.append(' ')
    preds_text.append(' '.join(temp))
```

In [19]:
```
#LET'S PUT THE ORIGINAL ENGLISH SENTENCES IN THE TEST DATASET AND THE PREDICTED SEQ
```

In [20]:
```
pred_df = pd.DataFrame({'actual': test[:, 0], 'predicted': preds_text})
```

In [21]:
```
#print 15 rows randomly
pred_df.sample(15, replace=True)
```

Out[21]:

	actual	predicted
3819	lets have a good time	i is a
2347	youre very observant	youre you
2188	tom has green eyes	i is a
1321	i have to stop	i is a
2162	thats a big deal	i is a
6176	is this love	i you
5349	tie your shoe	i you
7678	we cant save everyone	i not a to
8470	you must stop him	youre you
5500	life is too short	i is a
9822	this is my family	i is
6470	whats wrong with you	i is a
459	she has gone shopping	i not to
1601	im not a criminal	i not a to
3587	i think we should quit	i not a to

In [25]:
```
import string
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, RepeatVector
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

# Load and preprocess the data (similar to the provided code)

# Tokenization (similar to the provided code)

# Data encoding (similar to the provided code)

# Define the NMT model (similar to the provided code)

# Compile the model (similar to the provided code)

# Train the model (similar to the provided code)

# Take user input
user_input = "comment allez-vous"  # Enter the input in French

# Preprocess user input
user_input = user_input.lower()
user_input = user_input.translate(str.maketrans('', '', string.punctuation))
```

```
user_input_sequence = fra_tokenizer.texts_to_sequences([user_input])
user_input_sequence = pad_sequences(user_input_sequence, maxlen=fra_length, padding

# Translate user input
predicted_sequence = model.predict(user_input_sequence)
translated_text = ""

for i in predicted_sequence[0]:
    word = eng_tokenizer.index_word.get(np.argmax(i))
    if word:
        translated_text += word + " "

# Display the translation
print("Translated: " + translated_text)
```

```
1/1 [==============================] - 0s 206ms/step
Translated: youre you
```

In [ ]: