In [1]:

```
!wget  http://www.manythings.org/anki/fra-eng.zip
```

```
--2021-07-21 12:03:01--  http://www.manythings.org/anki/fra-eng.zip
Resolving www.manythings.org (www.manythings.org)... 104.21.92.44, 172.67.186.54, 2606:4700:8d9e:7d56:ab64:120:b19e:a55c
Connecting to www.manythings.org (www.manythings.org)|104.21.92.44|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6451478 (6.2M) [application/zip]
Saving to: ‘fra-eng.zip’

fra-eng.zip         100%[===================>]   6.15M  2.40MB/s    in 2.6s

2021-07-21 12:03:04 (2.40 MB/s) - ‘fra-eng.zip’ saved [6451478/6451478]

checkdir:  cannot create extraction directory: /content/french-eng
           No such file or directory
```

In [13]:

```
#importing libraries
from __future__ import print_function

from keras.models import Model
from keras.layers import Input, LSTM, Dense
import numpy as np
```

In [14]:

```
batch_s = 64  # Batch size for training.
epochs = 100  # Number of epochs to train for.
latent_dimension = 256  # Latent dimensionality of the encoding space.
number_samples = 10000  # Number of samples to train on.
# Path to the data txt file on disk.
data_path = 'fra.txt'
```

In [15]:

```
# Vectorize the data.
input_texts = []
target_texts = []
input_characters = set()
target_characters = set()
with open(data_path, 'r', encoding='utf-8') as f:
    lines = f.read().split('\n')
for line in lines[: min(number_samples, len(lines) - 1)]:
    input_text, target_text, _ = line.split('\t')
    # We use "tab" as the "start sequence" character
    # for the targets, and "\n" as "end sequence" character.
    target_text = '\t' + target_text + '\n'
    input_texts.append(input_text)
    target_texts.append(target_text)
    for char in input_text:
        if char not in input_characters:
            input_characters.add(char)
    for char in target_text:
        if char not in target_characters:
            target_characters.add(char)

input_characters = sorted(list(input_characters))
target_characters = sorted(list(target_characters))
num_encoder_tokens = len(input_characters)
num_decoder_tokens = len(target_characters)
max_encoder_seq_length = max([len(txt) for txt in input_texts])
max_decoder_seq_length = max([len(txt) for txt in target_texts])
```

In [17]:

```
print('Number of samples:', len(target_texts))
print('Number of unique input tokens:', num_encoder_tokens)
print('Number of unique output tokens:', num_decoder_tokens)
print('Max sequence length for inputs:', max_encoder_seq_length)
print('Max sequence length for outputs:', max_decoder_seq_length)
```

```
Number of samples: 10000
Number of unique input tokens: 70
Number of unique output tokens: 93
Max sequence length for inputs: 16
Max sequence length for outputs: 59
```

In [23]:

```
input_token_index = dict(
    [(char, i) for i, char in enumerate(input_characters)])
target_token_index = dict(
    [(char, i) for i, char in enumerate(target_characters)])
```

In [27]:

```
encoder_input_data = np.zeros(
    (len(input_texts), max_encoder_seq_length, num_encoder_tokens),
    dtype='float32')
decoder_input_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens),
    dtype='float32')
decoder_target_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens),
    dtype='float32')
```

In [12]:

```
for i, (input_text, target_text) in enumerate(zip(input_texts, target_texts)):
    for t, char in enumerate(input_text):
        encoder_input_data[i, t, input_token_index[char]] = 1.
    encoder_input_data[i, t + 1:, input_token_index[' ']] = 1.
    for t, char in enumerate(target_text):
        # decoder_target_data is ahead of decoder_input_data by one timestep
        decoder_input_data[i, t, target_token_index[char]] = 1.
        if t > 0:
            # decoder_target_data will be ahead by one timestep
            # and will not include the start character.
            decoder_target_data[i, t - 1, target_token_index[char]] = 1.
    decoder_input_data[i, t + 1:, target_token_index[' ']] = 1.
    decoder_target_data[i, t:, target_token_index[' ']] = 1.
```

In [28]:

```
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dimension, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

In [31]:

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dimension, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs,
                                     initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

In [32]:

```
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

In [33]:

```
# Run training
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
              metrics=['accuracy'])
#model.fit([encoder_input_data, decoder_input_data], decoder_target_data,
          #batch_size=batch_s,
          #epochs=epochs,
          #validation_split=0.2)
```

In [13]:

```
# Save model
model.save('seq2seq.h5')
```

In [15]:

```
# Next: inference mode (sampling).
# Here's the drill:
# 1) encode input and retrieve initial decoder state
# 2) run one step of decoder with this initial state
# and a "start of sequence" token as target.
# Output will be the next target token
# 3) Repeat with the current target token and current states

# Define sampling models
encoder_model = Model(encoder_inputs, encoder_states)

decoder_state_input_h = Input(shape=(latent_dimension,))
decoder_state_input_c = Input(shape=(latent_dimension,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs] + decoder_states)
```

In [16]:

```
# Reverse-lookup token index to decode sequences back to
# something readable.
reverse_input_char_index = dict(
    (i, char) for char, i in input_token_index.items())
reverse_target_char_index = dict(
    (i, char) for char, i in target_token_index.items())
```

In [17]:

```
def decode_sequence(input_seq):
    # Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)

    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1, 1, num_decoder_tokens))
    # Populate the first character of target sequence with the start character.
    target_seq[0, 0, target_token_index['\t']] = 1.

    # Sampling loop for a batch of sequences
    # (to simplify, here we assume a batch of size 1).
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict(
            [target_seq] + states_value)

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = reverse_target_char_index[sampled_token_index]
        decoded_sentence += sampled_char

        # Exit condition: either hit max length
        # or find stop character.
        if (sampled_char == '\n' or
           len(decoded_sentence) > max_decoder_seq_length):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1, 1, num_decoder_tokens))
        target_seq[0, 0, sampled_token_index] = 1.

        # Update states
        states_value = [h, c]

    return decoded_sentence
```

In [18]:

```
for seq_index in range(100):
    # Take one sequence (part of the training set)
    # for trying out decoding.
    input_seq = encoder_input_data[seq_index: seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    print('-')
    print('Input sentence:', input_texts[seq_index])
    print('Decoded sentence:', decoded_sentence)
```

```
-
Input sentence: Go.
Decoded sentence: Va !

-
Input sentence: Hi.
Decoded sentence: Salut.

-
Input sentence: Hi.
Decoded sentence: Salut.

-
Input sentence: Run!
Decoded sentence: Courez !

-
Input sentence: Run!
Decoded sentence: Courez !

-
Input sentence: Who?
Decoded sentence: Qui ?

-
Input sentence: Wow!
Decoded sentence: Ça alors !

-
Input sentence: Fire!
Decoded sentence: Au fim !

-
Input sentence: Help!
Decoded sentence: À l'aide !

-
Input sentence: Jump.
Decoded sentence: Saute.

-
Input sentence: Stop!
Decoded sentence: Arrête-toi !

-
Input sentence: Stop!
Decoded sentence: Arrête-toi !

-
Input sentence: Stop!
Decoded sentence: Arrête-toi !

-
Input sentence: Wait!
Decoded sentence: Attends !

-
Input sentence: Wait!
Decoded sentence: Attends !

-
Input sentence: Go on.
Decoded sentence: Poursuis.

-
Input sentence: Go on.
Decoded sentence: Poursuis.

-
Input sentence: Go on.
Decoded sentence: Poursuis.

-
Input sentence: Hello!
Decoded sentence: Salut !

-
Input sentence: Hello!
Decoded sentence: Salut !

-
Input sentence: I see.
Decoded sentence: Je comprends.

-
Input sentence: I try.
Decoded sentence: J'essaye.

-
Input sentence: I won!
Decoded sentence: Je l'ai emporté !

-
Input sentence: I won!
Decoded sentence: Je l'ai emporté !

-
Input sentence: I won.
Decoded sentence: J'ai gagné.

-
Input sentence: Oh no!
Decoded sentence: Oh non !

-
Input sentence: Attack!
Decoded sentence: Attaque !

-
Input sentence: Attack!
Decoded sentence: Attaque !

-
Input sentence: Cheers!
Decoded sentence: À votre santé !

-
Input sentence: Cheers!
Decoded sentence: À votre santé !

-
Input sentence: Cheers!
Decoded sentence: À votre santé !

-
Input sentence: Cheers!
Decoded sentence: À votre santé !

-
Input sentence: Get up.
Decoded sentence: Lève-toi.

-
Input sentence: Go now.
Decoded sentence: Va, maintenant.

-
Input sentence: Go now.
Decoded sentence: Va, maintenant.

-
Input sentence: Go now.
Decoded sentence: Va, maintenant.

-
Input sentence: Got it!
Decoded sentence: Compris !

-
Input sentence: Got it!
Decoded sentence: Compris !

-
Input sentence: Got it?
Decoded sentence: T'as capté ?

-
Input sentence: Got it?
Decoded sentence: T'as capté ?

-
Input sentence: Got it?
Decoded sentence: T'as capté ?

-
Input sentence: Hop in.
Decoded sentence: Montez.

-
Input sentence: Hop in.
Decoded sentence: Montez.

-
Input sentence: Hug me.
Decoded sentence: Serre-moi dans tes bras !

-
Input sentence: Hug me.
Decoded sentence: Serre-moi dans tes bras !

-
Input sentence: I fell.
Decoded sentence: Je suis tombé.

-
Input sentence: I fell.
Decoded sentence: Je suis tombé.

-
Input sentence: I know.
Decoded sentence: Je sais.

-
Input sentence: I left.
Decoded sentence: Je suis parti.

-
Input sentence: I left.
Decoded sentence: Je suis parti.

-
Input sentence: I lied.
Decoded sentence: J'ai menti.

-
Input sentence: I lost.
Decoded sentence: J'ai perdu.

-
Input sentence: I paid.
Decoded sentence: J’ai payé.

-
Input sentence: I'm 19.
Decoded sentence: J'ai exagé.

-
Input sentence: I'm OK.
Decoded sentence: Je vais bien.

-
Input sentence: I'm OK.
Decoded sentence: Je vais bien.

-
Input sentence: Listen.
Decoded sentence: Écoutez !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: No way!
Decoded sentence: C'est exclu !

-
Input sentence: Really?
Decoded sentence: Vraiment ?

-
Input sentence: Really?
Decoded sentence: Vraiment ?

-
Input sentence: Really?
Decoded sentence: Vraiment ?

-
Input sentence: Thanks.
Decoded sentence: Merci !

-
Input sentence: We try.
Decoded sentence: On essaye.

-
Input sentence: We won.
Decoded sentence: Nous gagnâmes.

-
Input sentence: We won.
Decoded sentence: Nous gagnâmes.

-
Input sentence: We won.
Decoded sentence: Nous gagnâmes.

-
Input sentence: We won.
Decoded sentence: Nous gagnâmes.

-
Input sentence: Ask Tom.
Decoded sentence: Demande à Tom.

-
Input sentence: Awesome!
Decoded sentence: Fantastique !

-
Input sentence: Be calm.
Decoded sentence: Soyez calmes !

-
Input sentence: Be calm.
Decoded sentence: Soyez calmes !

-
Input sentence: Be calm.
Decoded sentence: Soyez calmes !

-
Input sentence: Be cool.
Decoded sentence: Sois détendu !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be fair.
Decoded sentence: Soyez équitable !

-
Input sentence: Be kind.
Decoded sentence: Sois gentil.

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Be nice.
Decoded sentence: Soyez gentils !

-
Input sentence: Beat it.
Decoded sentence: Faiglez ça.

-
Input sentence: Call me.
Decoded sentence: Appelle-moi !

-
Input sentence: Call me.
Decoded sentence: Appelle-moi !

-
Input sentence: Call us.
Decoded sentence: Appelle-nous !

-
Input sentence: Call us.
Decoded sentence: Appelle-nous !

-
Input sentence: Come in.
Decoded sentence: Entrez !
```

In [0]: