In [2]:

```
!gdown --id 1OurDQUtbWQacvT32HMqFL7vIUrSMllOp
```

```
Downloading...
From: https://drive.google.com/uc?id=1OurDQUtbWQacvT32HMqFL7vIUrSMllOp
To: /content/preprocessed_data.csv
100% 300k/300k [00:00<00:00, 2.70MB/s]
```

In [3]:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [4]:

```
df=pd.read_csv('preprocessed_data.csv')
```

In [5]:

```
df.head(4)
```

Out[5]:

	Unnamed: 0	source	target
0	0	U wan me to "chop" seat 4 u nt?\n	Do you want me to reserve seat for you or not?\n
1	1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	3	I'm thai. what do u do?\n	I'm Thai. What do you do?\n

In [6]:

```
def preprocess(x):
  x=x[:-1]
  return x
```

In [7]:

```
df['source']=df['source'].apply(preprocess)
df['target']=df['target'].apply(preprocess)
```

In [8]:

```
df=df[['source','target']]
df.head()
```

Out[8]:

	source	target
0	U wan me to "chop" seat 4 u nt?	Do you want me to reserve seat for you or not?
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	I'm Thai. What do you do?
4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...

In [9]:

```
df.shape
```

Out[9]:

```
(2000, 2)
```

In [10]:

```
def length(text):#for calculating the length of the sentence
    return len(str(text))
```

In [11]:

```
df=df[df['source'].apply(length)<170]
df=df[df['target'].apply(length)<200]
```

In [12]:

```
df.shape
```

Out[12]:

```
(1990, 2)
```

In [12]:

```
df['target_in'] = '<start> ' + df['target'].astype(str)
df['target_out'] = df['target'].astype(str) + ' <end>'
```

```
# only for the first sentance add a toke <end> so that we will have <end> in tokenizer
df.head()
```

Out[12]:

	source	target	target_in	target_out
0	U wan me to "chop" seat 4 u nt?	Do you want me to reserve seat for you or not?	<start> Do you want me to reserve seat for you...	Do you want me to reserve seat for you or not?...
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...	<start> Yeap. You reaching? We ordered some Du...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...	<start> They become more expensive already. Mi...	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	I'm Thai. What do you do?	<start> I'm Thai. What do you do?	I'm Thai. What do you do? <end>
4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...	<start> Hi! How did your week go? Haven't hear...	Hi! How did your week go? Haven't heard from y...

In [13]:

```
df=df.drop('target',axis=1)
```

In [14]:

```
df.head(4)
```

Out[14]:

	source	target_in	target_out
0	U wan me to "chop" seat 4 u nt?	<start> Do you want me to reserve seat for you...	Do you want me to reserve seat for you or not?...
1	Yup. U reaching. We order some durian pastry a...	<start> Yeap. You reaching? We ordered some Du...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	<start> They become more expensive already. Mi...	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	<start> I'm Thai. What do you do?	I'm Thai. What do you do? <end>

In [15]:

```
from sklearn.model_selection import train_test_split
train, validation = train_test_split(df, test_size=0.01)
```

In [16]:

```
print(train.shape, validation.shape)
# for one sentence we will be adding <end> token so that the tokanizer learns the word <end>
# with this we can use only one tokenizer for both encoder output and decoder output
train.iloc[0]['target_in']= str(train.iloc[0]['target_in'])+' <end>'
train.iloc[0]['target_out']= str(train.iloc[0]['target_out'])+' <end>'
```

```
(1970, 3) (20, 3)
```

In [17]:

```
tknizer_source = Tokenizer()
tknizer_source.fit_on_texts(train['source'].values)
tknizer_target = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n')
tknizer_target.fit_on_texts(train['target_in'].values)
```

In [18]:

```
vocab_size_target=len(tknizer_target.word_index.keys())
print(vocab_size_target)
vocab_size_source=len(tknizer_source.word_index.keys())
print(vocab_size_source)
```

```
3039
3708
```

In [19]:

```
tknizer_target.word_index['<start>'], tknizer_target.word_index['<end>']
```

Out[19]:

```
(1, 1447)
```

In [20]:

```
class Encoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns output sequence
    '''

    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):

        #Initialize Embedding layer
        #Intialize Encoder LSTM layer
        super().__init__()
```

```
        self.vocab_size = inp_vocab_size
        self.embedding_size = embedding_size
        self.input_length = input_length
        self.lstm_size= lstm_size
        self.lstm_output=0
        self.embedding = tf.keras.layers.Embedding(input_dim=self.vocab_size, output_dim=self.embedding_s
                           mask_zero=True, name="embedding_layer_encoder")
        self.lstm = tf.keras.layers.LSTM(self.lstm_size, return_state=True, return_sequences=True, name="


    def call(self,input_sequence,states):

      '''
          This function takes a sequence input and the initial states of the encoder.
          Pass the input_sequence input to the Embedding layer, Pass the embedding layer oupulstm_state_h
          returns -- All encoder_outputs, last time steps hidden and cell state
      '''

      input_embedd                             = self.embedding(input_sequence)
      lstm_state_h,lstm_state_c= states[0],states[1]
      self.lstm_output,lstm_state_h,lstm_state_c=self.lstm(input_embedd)
      return self.lstm_output,lstm_state_h,lstm_state_c



    def initialize_states(self,batch_size):
      '''
      Given a batch size it will return intial hidden state and intial cell state.
      If batch size is 32- Hidden state is zeros of size [32,lstm_units], cell state zeros is of size [32
      '''
      return [tf.zeros((batch_size,self.lstm_size)),tf.zeros((batch_size,self.lstm_size))]
```

In [21]:

```
class Attention(tf.keras.layers.Layer):
  '''
    Class the calculates score based on the scoring_function using Bahdanu attention mechanism.
  '''
  def __init__(self,scoring_function, att_units):
    super().__init__()
    self.scoring_function=scoring_function
    # Please go through the reference notebook and research paper to complete the scoring functions

    if self.scoring_function=='dot':
      # Intialize variables needed for Dot score function here
      pass
    if self.scoring_function == 'general':
      # Intialize variables needed for General score function here
      self.weight=tf.keras.layers.Dense(att_units)
    elif self.scoring_function == 'concat':
      # Intialize variables needed for Concat score function here
      self.weight1=tf.keras.layers.Dense(att_units)
      self.weight2=tf.keras.layers.Dense(att_units)
      self.v=tf.keras.layers.Dense(1)


  def call(self,decoder_hidden_state,encoder_output):
    '''
      Attention mechanism takes two inputs current step -- decoder_hidden_state and all the encoder_outpu
      * Based on the scoring function we will find the score or similarity between decoder_hidden_state a
        Multiply the score function with your encoder_outputs to get the context vector.
        Function returns context vector and attention weights(softmax - scores)
    '''

    if self.scoring_function == 'dot':
        # Implement Dot score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=2)
        value=tf.matmul(encoder_output,decoder_hidden_state)
    elif self.scoring_function == 'general':
        # Implement General score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=2)
        value=tf.matmul(self.weight(encoder_output),decoder_hidden_state)
    elif self.scoring_function == 'concat':
        # Implement General score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=1)
        value=self.v(tf.nn.tanh(self.weight1(decoder_hidden_state)+self.weight2(encoder_output)))
    attention_weights=tf.nn.softmax(value,axis=1)
```

```
    context_vector=attention_weights*encoder_output
    return tf.reduce_sum(context_vector,axis=1),attention_weights
```

In [22]:

```
class One_Step_Decoder(tf.keras.Model):
  def __init__(self,tar_vocab_size, embedding_dim, input_length, dec_units ,score_fun ,att_units):

      # Initialize decoder embedding layer, LSTM and any other objects needed
      super().__init__()
      self.tar_vocab_size = tar_vocab_size
      self.embedding_dim = embedding_dim
      self.input_length = input_length
      self.dec_units = dec_units
      self.score_fun = score_fun
      self.att_units = att_units
      # we are using embedding_matrix and not training the embedding layer
      self.embedding = tf.keras.layers.Embedding(input_dim=self.tar_vocab_size, output_dim=self.embedding
                          mask_zero=True, name="embedding_layer_decoder", trainable=True)
      self.lstm = tf.keras.layers.LSTM(self.dec_units, return_sequences=True, return_state=True)
      self.dense = tf.keras.layers.Dense(self.tar_vocab_size)
      self.attention = Attention(self.score_fun,self.att_units)

  def call(self,input_to_decoder, encoder_output, state_h,state_c):
    '''
        One step decoder mechanisim step by step:
      A. Pass the input_to_decoder to the embedding layer and then get the output(batch_size,1,embedding_
      B. Using the encoder_output and decoder hidden state, compute the context vector.
      C. Concat the context vector with the step A output
      D. Pass the Step-C output to LSTM/GRU and get the decoder output and states(hidden and cell state)
      E. Pass the decoder output to dense layer(vocab size) and store the result into output.
      F. Return the states from step D, output from Step E, attention weights from Step -B
    '''
    output = self.embedding(input_to_decoder)
    context_vector,attention_weights = self.attention(state_h,encoder_output)
    context_vector1 = tf.expand_dims(context_vector,1)
    concat = tf.concat([output,context_vector1],axis=-1)
    decoder_output,state_h,state_c = self.lstm(concat,initial_state=[state_h,state_c])
    final_output = self.dense(decoder_output)
    final_output = tf.reshape(final_output,(-1,final_output.shape[2]))
    return final_output,state_h,state_c,attention_weights,context_vector
```

In [23]:

```
class Decoder(tf.keras.Model):
    def __init__(self,out_vocab_size, embedding_dim, input_length, dec_units ,score_fun ,att_units):
      #Intialize necessary variables and create an object from the class onestepdecoder
      super(Decoder,self).__init__()
      self.vocab_size = out_vocab_size
      self.embedding_dim = embedding_dim
      self.input_length = input_length
      self.dec_units=dec_units
      self.att_units=att_units
      self.score_fun=score_fun
      self.onestepdecoder=One_Step_Decoder(self.vocab_size,self.embedding_dim,self.input_length,self.dec_

    def call(self, input_to_decoder,encoder_output,decoder_hidden_state,decoder_cell_state ):

        #Initialize an empty Tensor array, that will store the outputs at each and every time step
        all_outputs=tf.TensorArray(tf.float32,size=input_to_decoder.shape[1])
        #Create a tensor array as shown in the reference notebook
        #Iterate till the length of the decoder input
        for timestep in range(input_to_decoder.shape[1]):
            # Call onestepdecoder for each token in decoder_input
            output,state_h,state_c,attention_weights,context_vector=self.onestepdecoder(input_to_decoder[

            # Store the output in tensorarray
            all_outputs=all_outputs.write(timestep,output)
        all_outputs=tf.transpose(all_outputs.stack(),[1,0,2])
        # Return the tensor array
        return all_outputs
```

In [24]:

```
class encoder_decoder(tf.keras.Model):
```

```
  def __init__(self,encoder_inputs_length,decoder_inputs_length, output_vocab_size,batch_size,score_fun):
    #Intialize objects from encoder decoder
    super().__init__() # https://stackoverflow.com/a/27134600/4084039
    self.batch_size=batch_size
    self.encoder = Encoder(vocab_size_source+1,300,128,encoder_inputs_length)
    self.decoder = Decoder(vocab_size_target+1,300,decoder_inputs_length,128,score_fun,128)

  def call(self,data):
    #Intialize encoder states, Pass the encoder_sequence to the embedding layer
    # Decoder initial states are encoder final states, Initialize it accordingly
    # Pass the decoder sequence,encoder_output,decoder states to Decoder
    # return the decoder output
    input,output = data[0], data[1]
    initial_state=self.encoder.initialize_states(self.batch_size)
    encoder_output, encoder_h, encoder_c = self.encoder(input,initial_state)
    decoder_output= self.decoder(output, encoder_output, encoder_h, encoder_c)
    return decoder_output
```

In [25]:

```
#https://www.tensorflow.org/tutorials/text/image_captioning#model
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')
def loss_function(real, pred):
    """ Custom loss function that will not consider the loss for padded zeros.
    why are we using this, can't we use simple sparse categorical crossentropy?
    Yes, you can use simple sparse categorical crossentropy as loss like we did in task-1. But in this lo
    for the padded zeros. i.e when the input is zero then we donot need to worry what the output is. This
    during preprocessing to make equal length for all the sentences.

    """


    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)

    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask

    return tf.reduce_mean(loss_)
```

In [26]:

```
class Dataset:
    def __init__(self, df, tknizer_source, tknizer_target, source_len,target_len):
        self.encoder_inps = df['source'].values
        self.decoder_inps = df['target_in'].values
        self.decoder_outs = df['target_out'].values
        self.tknizer_target = tknizer_target
        self.tknizer_source = tknizer_source
        self.source_len = source_len
        self.target_len = target_len


    def __getitem__(self, i):
        self.encoder_seq = self.tknizer_source.texts_to_sequences([self.encoder_inps[i]]) # need to pass
        self.decoder_inp_seq = self.tknizer_target.texts_to_sequences([self.decoder_inps[i]])
        self.decoder_out_seq = self.tknizer_target.texts_to_sequences([self.decoder_outs[i]])

        self.encoder_seq = pad_sequences(self.encoder_seq, maxlen=self.source_len, dtype='int32', padding
        self.decoder_inp_seq = pad_sequences(self.decoder_inp_seq, maxlen=self.target_len, dtype='int32',
        self.decoder_out_seq = pad_sequences(self.decoder_out_seq, maxlen=self.target_len, dtype='int32',
        return self.encoder_seq, self.decoder_inp_seq, self.decoder_out_seq

    def __len__(self): # your model.fit_gen requires this function
        return len(self.encoder_inps)


class Dataloder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size=1):
        self.dataset = dataset
        self.batch_size = batch_size
        self.indexes = np.arange(len(self.dataset.encoder_inps))


    def __getitem__(self, i):
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
```

```
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.squeeze(np.stack(samples, axis=1), axis=0) for samples in zip(*data)]
        # we are creating data like ([italian, english_inp], english_out) these are already converted in
        return tuple([[batch[0],batch[1]],batch[2]])

    def __len__(self):  # your model.fit_gen requires this function
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        self.indexes = np.random.permutation(self.indexes)
```

In [27]:

```
train_dataset = Dataset(train, tknizer_source, tknizer_target,39,43)
test_dataset  = Dataset(validation, tknizer_source, tknizer_target,39,43)

train_dataloader = Dataloder(train_dataset, batch_size=512)
test_dataloader = Dataloder(test_dataset, batch_size=20)


print(train_dataloader[0][0][0].shape, train_dataloader[0][0][1].shape, train_dataloader[0][1].shape)
```

```
(512, 39) (512, 43) (512, 43)
```

In [28]:

```
tf.config.experimental_run_functions_eagerly(True)
```

```
WARNING:tensorflow:From <ipython-input-28-bdb3352f611a>:1: experimental_run_functions_eagerly (from tenso
rflow.python.eager.def_function) is deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.config.run_functions_eagerly` instead of the experimental version.
```

In [29]:

```
tf.config.run_functions_eagerly(True)
```

In [30]:

```
#Create an object of encoder_decoder Model class,
# Compile the model and fit the model
# Implement teacher forcing while training your model. You can do it two ways.
# Prepare your data, encoder_input,decoder_input and decoder_output
# if decoder input is
# <start> Hi how are you
# decoder output should be
# Hi How are you <end>
# i.e when you have send <start>-- decoder predicted Hi, 'Hi' decoder predicted 'How' .. e.t.c

# or

# model.fit([train_ita,train_eng],train_eng[:,1:]..)
# Note: If you follow this approach some grader functions might return false and this is fine.
model  = encoder_decoder(encoder_inputs_length=39,decoder_inputs_length=43,output_vocab_size=vocab_size_t
optimizer = tf.keras.optimizers.Adam(0.01)
model.compile(optimizer=optimizer,loss=loss_function)
train_steps=train.shape[0]//512
valid_steps=validation.shape[0]//20
model.fit_generator(train_dataloader, steps_per_epoch=train_steps, epochs=100, validation_data=test_datal
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1940: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops/dataset_ops.py:3704: UserWarning: Even
though the `tf.config.experimental_run_functions_eagerly` option is set, this option does not apply to
tf.data functions. To force eager execution of tf.data functions, please use
`tf.data.experimental.enable.debug_mode()`.
  "Even though the `tf.config.experimental_run_functions_eagerly` "
Epoch 1/100
3/3 [==============================] - 4s 1s/step - loss: 2.7480 - val_loss: 2.1525
Epoch 2/100
3/3 [==============================] - 3s 1s/step - loss: 2.3138 - val_loss: 1.6742
Epoch 3/100
3/3 [==============================] - 3s 1s/step - loss: 2.1388 - val_loss: 1.6743
Epoch 4/100
3/3 [==============================] - 3s 1s/step - loss: 2.1127 - val_loss: 1.6691
Epoch 5/100
3/3 [==============================] - 3s 1s/step - loss: 2.0877 - val_loss: 1.6687
Epoch 6/100
3/3 [==============================] - 3s 997ms/step - loss: 2.0770 - val_loss: 1.6678
```

```
Epoch 7/100
3/3 [==============================] - 3s 1s/step - loss: 2.0696 - val_loss: 1.6660
Epoch 8/100
3/3 [==============================] - 3s 1s/step - loss: 2.0689 - val_loss: 1.6673
Epoch 9/100
3/3 [==============================] - 3s 1s/step - loss: 2.0653 - val_loss: 1.6610
Epoch 10/100
3/3 [==============================] - 3s 1s/step - loss: 2.0611 - val_loss: 1.6525
Epoch 11/100
3/3 [==============================] - 3s 1s/step - loss: 2.0554 - val_loss: 1.6501
Epoch 12/100
3/3 [==============================] - 3s 1s/step - loss: 2.0531 - val_loss: 1.6501
Epoch 13/100
3/3 [==============================] - 3s 1s/step - loss: 2.0526 - val_loss: 1.6506
Epoch 14/100
3/3 [==============================] - 3s 1s/step - loss: 2.0467 - val_loss: 1.6473
Epoch 15/100
3/3 [==============================] - 3s 995ms/step - loss: 2.0405 - val_loss: 1.6434
Epoch 16/100
3/3 [==============================] - 3s 1s/step - loss: 2.0412 - val_loss: 1.6406
Epoch 17/100
3/3 [==============================] - 3s 1s/step - loss: 2.0362 - val_loss: 1.6326
Epoch 18/100
3/3 [==============================] - 3s 1s/step - loss: 2.0259 - val_loss: 1.6242
Epoch 19/100
3/3 [==============================] - 3s 1s/step - loss: 2.0170 - val_loss: 1.6191
Epoch 20/100
3/3 [==============================] - 3s 1s/step - loss: 2.0039 - val_loss: 1.6164
Epoch 21/100
3/3 [==============================] - 3s 1s/step - loss: 1.9921 - val_loss: 1.6097
Epoch 22/100
3/3 [==============================] - 3s 1s/step - loss: 1.9881 - val_loss: 1.6019
Epoch 23/100
3/3 [==============================] - 3s 1s/step - loss: 1.9725 - val_loss: 1.5966
Epoch 24/100
3/3 [==============================] - 3s 1s/step - loss: 1.9652 - val_loss: 1.5874
Epoch 25/100
3/3 [==============================] - 3s 1s/step - loss: 1.9470 - val_loss: 1.5802
Epoch 26/100
3/3 [==============================] - 3s 1s/step - loss: 1.9339 - val_loss: 1.5761
Epoch 27/100
3/3 [==============================] - 3s 994ms/step - loss: 1.9240 - val_loss: 1.5676
Epoch 28/100
3/3 [==============================] - 3s 1s/step - loss: 1.9093 - val_loss: 1.5616
Epoch 29/100
3/3 [==============================] - 3s 1s/step - loss: 1.8943 - val_loss: 1.5545
Epoch 30/100
3/3 [==============================] - 3s 1s/step - loss: 1.8795 - val_loss: 1.5468
Epoch 31/100
3/3 [==============================] - 3s 1s/step - loss: 1.8640 - val_loss: 1.5409
Epoch 32/100
3/3 [==============================] - 3s 1s/step - loss: 1.8551 - val_loss: 1.5359
Epoch 33/100
3/3 [==============================] - 3s 1s/step - loss: 1.8404 - val_loss: 1.5304
Epoch 34/100
3/3 [==============================] - 3s 1s/step - loss: 1.8232 - val_loss: 1.5223
Epoch 35/100
3/3 [==============================] - 3s 1s/step - loss: 1.8106 - val_loss: 1.5202
Epoch 36/100
3/3 [==============================] - 3s 1s/step - loss: 1.7934 - val_loss: 1.5148
Epoch 37/100
3/3 [==============================] - 3s 1s/step - loss: 1.7828 - val_loss: 1.5105
Epoch 38/100
3/3 [==============================] - 3s 1s/step - loss: 1.7660 - val_loss: 1.5016
Epoch 39/100
3/3 [==============================] - 3s 1s/step - loss: 1.7475 - val_loss: 1.5015
Epoch 40/100
3/3 [==============================] - 3s 1s/step - loss: 1.7397 - val_loss: 1.4986
Epoch 41/100
3/3 [==============================] - 3s 1s/step - loss: 1.7245 - val_loss: 1.4938
Epoch 42/100
3/3 [==============================] - 3s 1s/step - loss: 1.7091 - val_loss: 1.4948
Epoch 43/100
3/3 [==============================] - 3s 1s/step - loss: 1.6946 - val_loss: 1.4843
Epoch 44/100
3/3 [==============================] - 3s 1s/step - loss: 1.6806 - val_loss: 1.4825
Epoch 45/100
```

```
3/3 [==============================] - 3s 1s/step - loss: 1.6709 - val_loss: 1.4828
Epoch 46/100
3/3 [==============================] - 3s 1s/step - loss: 1.6554 - val_loss: 1.4794
Epoch 47/100
3/3 [==============================] - 3s 1s/step - loss: 1.6412 - val_loss: 1.4796
Epoch 48/100
3/3 [==============================] - 3s 1s/step - loss: 1.6257 - val_loss: 1.4754
Epoch 49/100
3/3 [==============================] - 3s 1s/step - loss: 1.6143 - val_loss: 1.4739
Epoch 50/100
3/3 [==============================] - 3s 1s/step - loss: 1.6050 - val_loss: 1.4752
Epoch 51/100
3/3 [==============================] - 3s 1s/step - loss: 1.5926 - val_loss: 1.4792
Epoch 52/100
3/3 [==============================] - 3s 1s/step - loss: 1.5776 - val_loss: 1.4750
Epoch 53/100
3/3 [==============================] - 3s 1s/step - loss: 1.5657 - val_loss: 1.4812
Epoch 54/100
3/3 [==============================] - 3s 1s/step - loss: 1.5504 - val_loss: 1.4821
Epoch 55/100
3/3 [==============================] - 3s 1s/step - loss: 1.5394 - val_loss: 1.4719
Epoch 56/100
3/3 [==============================] - 3s 1s/step - loss: 1.5263 - val_loss: 1.4826
Epoch 57/100
3/3 [==============================] - 3s 1s/step - loss: 1.5172 - val_loss: 1.4778
Epoch 58/100
3/3 [==============================] - 3s 1s/step - loss: 1.5080 - val_loss: 1.4764
Epoch 59/100
3/3 [==============================] - 3s 1s/step - loss: 1.4930 - val_loss: 1.4788
Epoch 60/100
3/3 [==============================] - 3s 1s/step - loss: 1.4879 - val_loss: 1.4797
Epoch 61/100
3/3 [==============================] - 3s 1s/step - loss: 1.4751 - val_loss: 1.4854
Epoch 62/100
3/3 [==============================] - ETA: 0s - loss: 1.4650
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-30-077b737c7789> in <module>()
     18 train_steps=train.shape[0]//512
     19 valid_steps=validation.shape[0]//20
---> 20 model.fit_generator(train_dataloader, steps_per_epoch=train_steps, epochs=100, validation_data=te
st_dataloader, validation_steps=valid_steps)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in fit_generator(self,
generator, steps_per_epoch, epochs, verbose, callbacks, validation_data, validation_steps,
validation_freq, class_weight, max_queue_size, workers, use_multiprocessing, shuffle, initial_epoch)
   1955         use_multiprocessing=use_multiprocessing,
   1956         shuffle=shuffle,
-> 1957         initial_epoch=initial_epoch)
   1958
   1959   def evaluate_generator(self,

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in fit(self, x, y, batc
h_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq,
max_queue_size, workers, use_multiprocessing)
   1223               use_multiprocessing=use_multiprocessing,
   1224               return_dict=True,
-> 1225               _use_cached_eval_dataset=True)
   1226           val_logs = {'val_' + name: val for name, val in val_logs.items()}
   1227           epoch_logs.update(val_logs)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in evaluate(self, x, y,
batch_size, verbose, sample_weight, steps, callbacks, max_queue_size, workers, use_multiprocessing,
return_dict, **kwargs)
   1487             with trace.Trace('test', step_num=step, _r=1):
   1488               callbacks.on_test_batch_begin(step)
-> 1489               tmp_logs = self.test_function(iterator)
   1490               if data_handler.should_sync:
   1491                 context.async_wait()

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in
test_function(iterator)
   1321       def test_function(iterator):
   1322         """Runs an evaluation execution with one step."""
-> 1323         return step_function(self, iterator)
   1324
```

```
   1325     else:

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in
step_function(model, iterator)
   1312
   1313       data = next(iterator)
-> 1314       outputs = model.distribute_strategy.run(run_step, args=(data,))
   1315       outputs = reduce_per_replica(
   1316           outputs, self.distribute_strategy, reduction='first')

/usr/local/lib/python3.7/dist-packages/tensorflow/python/distribute/distribute_lib.py in run(***failed re
solving arguments***)
   1283       fn = autograph.tf_convert(
   1284           fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
-> 1285       return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
   1286
   1287   def reduce(self, reduce_op, value, axis):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/distribute/distribute_lib.py in call_for_each_re
plica(self, fn, args, kwargs)
   2831       kwargs = {}
   2832     with self._container_strategy().scope():
-> 2833       return self._call_for_each_replica(fn, args, kwargs)
   2834
   2835   def _call_for_each_replica(self, fn, args, kwargs):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/distribute/distribute_lib.py in _call_for_each_r
eplica(self, fn, args, kwargs)
   3606   def _call_for_each_replica(self, fn, args, kwargs):
   3607     with ReplicaContext(self._container_strategy(), replica_id_in_sync_group=0):
-> 3608       return fn(*args, **kwargs)
   3609
   3610   def _reduce_to(self, reduce_op, value, destinations, options):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args,
**kwargs)
    595   def wrapper(*args, **kwargs):
    596     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.UNSPECIFIED):
--> 597       return func(*args, **kwargs)
    598
    599   if inspect.isfunction(func) or inspect.ismethod(func):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in run_step(data)
   1305
   1306       def run_step(data):
-> 1307         outputs = model.test_step(data)
   1308         # Ensure counter is updated only if `test_step` succeeds.
   1309         with ops.control_dependencies(_minimum_control_deps(outputs)):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py in test_step(self,
data)
   1264     x, y, sample_weight = data_adapter.unpack_x_y_sample_weight(data)
   1265
-> 1266     y_pred = self(x, training=False)
   1267     # Updates stateful loss metrics.
   1268     self.compiled_loss(

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py in __call__(self,
*args, **kwargs)
   1028         with autocast_variable.enable_auto_cast_variables(
   1029             self._compute_dtype_object):
-> 1030           outputs = call_fn(inputs, *args, **kwargs)
   1031
   1032         if self._activity_regularizer:

<ipython-input-24-ffdeee911166> in call(self, data)
     15     initial_state=self.encoder.initialize_states(self.batch_size)
     16     encoder_output, encoder_h, encoder_c = self.encoder(input,initial_state)
---> 17     decoder_output= self.decoder(output, encoder_output, encoder_h, encoder_c)
     18     return decoder_output
     19

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py in __call__(self,
*args, **kwargs)
   1028         with autocast_variable.enable_auto_cast_variables(
   1029             self._compute_dtype_object):
-> 1030           outputs = call_fn(inputs, *args, **kwargs)
```

```
   1031
   1032         if self._activity_regularizer:

<ipython-input-23-c3bfc02267d9> in call(self, input_to_decoder, encoder_output, decoder_hidden_state,
decoder_cell_state)
     19         for timestep in range(input_to_decoder.shape[1]):
     20             # Call onestepdecoder for each token in decoder_input
---> 21             output,state_h,state_c,attention_weights,context_vector=self.onestepdecoder(input_to_d
ecoder[:,timestep:timestep+1],encoder_output,decoder_hidden_state,decoder_cell_state)
     22
     23             # Store the output in tensorarray

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py in __call__(self,
*args, **kwargs)
   1028         with autocast_variable.enable_auto_cast_variables(
   1029             self._compute_dtype_object):
-> 1030           outputs = call_fn(inputs, *args, **kwargs)
   1031
   1032         if self._activity_regularizer:

<ipython-input-22-252336fceb78> in call(self, input_to_decoder, encoder_output, state_h, state_c)
     32     concat = tf.concat([output,context_vector1],axis=-1)
     33     decoder_output,state_h,state_c = self.lstm(concat,initial_state=[state_h,state_c])
---> 34     final_output = self.dense(decoder_output)
     35     final_output = tf.reshape(final_output,(-1,final_output.shape[2]))
     36     return final_output,state_h,state_c,attention_weights,context_vector

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/base_layer.py in __call__(self,
*args, **kwargs)
   1028         with autocast_variable.enable_auto_cast_variables(
   1029             self._compute_dtype_object):
-> 1030           outputs = call_fn(inputs, *args, **kwargs)
   1031
   1032         if self._activity_regularizer:

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/layers/core.py in call(self, inputs)
   1243     # Broadcast kernel to inputs.
   1244     else:
-> 1245       outputs = standard_ops.tensordot(inputs, self.kernel, [[rank - 1], [0]])
   1246       # Reshape the output back to the original ndim of the input.
   1247       if not context.executing_eagerly():

/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py in wrapper(*args, **kwargs)
    204     """Call target, and fall back on dispatchers if there is a TypeError."""
    205     try:
--> 206       return target(*args, **kwargs)
    207     except (TypeError, ValueError):
    208       # Note: convert_to_eager_tensor currently raises a ValueError, not a

/usr/local/lib/python3.7/dist-packages/tensorflow/python/ops/math_ops.py in tensordot(a, b, axes, name)
   4795   with ops.name_scope(name, "Tensordot", [a, b, axes]) as name:
   4796     a = ops.convert_to_tensor(a, name="a")
-> 4797     b = ops.convert_to_tensor(b, name="b")
   4798     a_axes, b_axes = _tensordot_axes(a, axes)
   4799     a_reshape, a_free_dims, a_free_dims_static = _tensordot_reshape(a, a_axes)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/profiler/trace.py in wrapped(*args, **kwargs)
    161         with Trace(trace_name, **trace_kwargs):
    162           return func(*args, **kwargs)
--> 163       return func(*args, **kwargs)
    164
    165     return wrapped

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in convert_to_tensor(value,
dtype, name, as_ref, preferred_dtype, dtype_hint, ctx, accepted_result_types)
   1549     # cast to preferred_dtype first.
   1550     ret = None
-> 1551     if dtype is None and preferred_dtype is not None:
   1552       try:
   1553         ret = conversion_func(

KeyboardInterrupt:
```

In [31]:

```
batch_size=512
units=128
```

In [40]:

```
def predict(input_sentence):

  '''
  A. Given input sentence, convert the sentence into integers using tokenizer used earlier
  B. Pass the input_sequence to encoder. we get encoder_outputs, last time step hidden and cell state
  C. Initialize index of <start> as input to decoder. and encoder final states as input_states to onestep
  D. till we reach max_length of decoder or till the model predicted word <end>:
        predictions, input_states, attention_weights = model.layers[1].onestepdecoder(input_to_decoder,
        Save the attention weights
        And get the word using the tokenizer(word index) and then store it in a string.
  E. Call plot_attention(#params)
  F. Return the predicted sentence
  '''
  initial_state_enc=[np.zeros((batch_size,units)),np.zeros((batch_size,units))]
  inp_seq = tknizer_source.texts_to_sequences([input_sentence])
  inp_seq = pad_sequences(inp_seq,padding='post',maxlen=39)

  en_outputs,state_h , state_c = model.layers[0](tf.constant(inp_seq),initial_state_enc)
  cur_vec = tf.constant([[tknizer_target.word_index['<start>']]])
  pred = []
  for i in range(43):
    output,state_h,state_c,attention_weights,context_vector = model.layers[1].onestepdecoder(cur_vec,en_o
    cur_vec = np.reshape(np.argmax(output), (1, 1))
    pred.append(tknizer_target.index_word[cur_vec[0][0]])
    if(pred[-1]=='<end>'):
      break
    translated_sentence = ' '.join(pred)
  return translated_sentence
```

In [41]:

```
for i in validation['source']:
  print("The Actual Output:")
  print(i)
  predicted=predict(i)
  print("The predicted output is: ")
  print(predicted)
  print('>'*100)
```

```
The Actual Output:
Helo k.reen n p.ple.hw r u?care 2 chat any1?
The predicted output is:
hi care to chat
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Haha... Not accurate right....
The predicted output is:
ok
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
oh cz1102? ya dat was me.i offline liao.im in clubrm ma
The predicted output is:
hey are you are you are you are you are you are you are you are you are you are you are you are you are you are you
you are you are you are you are you are you are you are you are you are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
eh dont think so... Library can print?
The predicted output is:
hey i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Sigh wat's new man... So tis is her no wat bf?
The predicted output is:
hey i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Hey its meijun's bday today so we share n treat her to crepes k.
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Sen, u male o female
The predicted output is:
ok
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Yupz... Okie... But u r always e bz one...Haha... K la, guess i'll cya at our class bbq den...
The predicted output is:
```

```
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
So boring, sending sms
The predicted output is:
hi care to chat
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Me very hungry... Ü come down faster lei...
The predicted output is:
hey i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
i am dying of boredom at home!i need a job!arghz!any recommendations?
The predicted output is:
hey are you are you are you are you are you are you are you are you are you are you are you are you are
you are you are you are you are you are you are you are you are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Pick me up at 6... Same place, car park there...
The predicted output is:
hey are you are you are you are you are you are you are you are you are you are you are you are you are
you are you are you are you are you are you are you are you are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Hey... R u free tmr? Wan to go for a movie? Cöz me not workin tmr... =5
The predicted output is:
hey i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Hey dont save seat for rebecca... She's not going for lect.
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Thanx.thats wat i aim to do.Lolx...Now u can't use me...Too bad...I need to sleep now.No matter how much
i wanna talk abt e sun moon n stars.Thanx 4
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Yun ah...Driving nid 2 bring é advance theory bk nt?
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Halo cy càrè for ä intro
The predicted output is:
hi care to chat
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Oh, hope ü have a good trip. Dun worry, i ll try hard not to miss you. Haha....  :-p
The predicted output is:
i am
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Rain,u busy nw? Y so quiet? Speak up.
The predicted output is:
hey are you are you are you are you are you are you are you are you are you are you are you are you are
you are you are you are you are you are you are you are you are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output:
Hey... You have driving lessons this weekend?
The predicted output is:
hey are you are you are you are you are you are you are you are you are you are you are you are you are
you are you are you are you are you are you are you are you are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [42]:

```
# Predict on 1000 random sentences on test data and calculate the average BLEU score of these sentences.
import nltk.translate.bleu_score as bleu
bleu_scores_lst=[]
for i in validation[:]['source']:
  reference = [i.split(),] # the original
  predicted=predict(i)
  translation = predicted.split()
```

```
  values=bleu.sentence_bleu(reference, translation)
  bleu_scores_lst.append(values)

# https://www.nltk.org/_modules/nltk/translate/bleu_score.html
```

```
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
```

In [43]:

```
average_bleu_scores=sum(bleu_scores_lst)/len(bleu_scores_lst)
print("Average BLEU score of these 20 test data sentences is: ",average_bleu_scores)
```

```
Average BLEU score of these 20 test data sentences is:  0.010138124061366445
```

In [44]:

```
bleu_scores_lst
```

Out[44]:

```
[0.2025894847023147,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 1.9007028956229227e-06,
 0,
 0,
 0.00017109582211857066,
 0,
 0]
```

Character_Level:

In [13]:

```
df=pd.read_csv('preprocessed_data.csv')
df.head()
```

Out[13]:

	Unnamed: 0	source	target
0	0	U wan me to "chop" seat 4 u nt?\n	Do you want me to reserve seat for you or not?\n
1	1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	3	I'm thai. what do u do?\n	I'm Thai. What do you do?\n
4	4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...

In [14]:

```
def preprocess(x):
  x=x[:-1]
  return x
```

In [15]:

```
df['source']=df['source'].apply(preprocess)
df['target']=df['target'].apply(preprocess)
```

In [16]:

```
df=df[['source','target']]
df.head()
```

Out[16]:

	source	target
0	U wan me to "chop" seat 4 u nt?	Do you want me to reserve seat for you or not?
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	I'm Thai. What do you do?
4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...

In [17]:

```
df.shape
```

Out[17]:

```
(2000, 2)
```

In [18]:

```
def length(text):#for calculating the length of the sentence
    return len(str(text))
```

In [19]:

```
df=df[df['source'].apply(length)<170]
df=df[df['target'].apply(length)<200]
```

In [20]:

```
df.shape
```

Out[20]:

```
(1990, 2)
```

In [21]:

```
df['target_in'] = '\t' + df['target'].astype(str)
df['target_out'] = df['target'].astype(str) + '\n'
# only for the first sentance add a toke <end> so that we will have <end> in tokenizer
df.head()
```

Out[21]:

	source	target	target_in	target_out
0	U wan me to "chop" seat 4 u nt?	Do you want me to reserve seat for you or not?	\tDo you want me to reserve seat for you or not?	Do you want me to reserve seat for you or not?\n
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...	\tYeap. You reaching? We ordered some Durian p...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...	\tThey become more expensive already. Mine is ...	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	I'm Thai. What do you do?	\tI'm Thai. What do you do?	I'm Thai. What do you do?\n
4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...	\tHi! How did your week go? Haven't heard from...	Hi! How did your week go? Haven't heard from y...

In [22]:

```
df=df.drop('target',axis=1)
```

In [23]:

```
df.head(4)
```

Out[23]:

	source	target_in	target_out
0	U wan me to "chop" seat 4 u nt?	\tDo you want me to reserve seat for you or not?	Do you want me to reserve seat for you or not?\n
1	Yup. U reaching. We order some durian pastry a...	\tYeap. You reaching? We ordered some Durian p...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	\tThey become more expensive already. Mine is ...	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	\tI'm Thai. What do you do?	I'm Thai. What do you do?\n

In [24]:

```
from sklearn.model_selection import train_test_split
train, validation = train_test_split(df, test_size=0.01)
```

In [25]:

```
print(train.shape, validation.shape)
# for one sentence we will be adding <end> token so that the tokanizer learns the word <end>
# with this we can use only one tokenizer for both encoder output and decoder output
```

```
train.iloc[0]['target_in']= str(train.iloc[0]['target_in'])+'\n'
train.iloc[0]['target_out']= str(train.iloc[0]['target_out'])+'\n'
```

```
(1970, 3) (20, 3)
```

In [26]:

```
tknizer_source = Tokenizer(filters=None,char_level=True,lower=False)
tknizer_source.fit_on_texts(train['source'].values)
tknizer_target = Tokenizer(filters=None,char_level=True,lower=False)
tknizer_target.fit_on_texts(train['target_in'].values)
```

In [27]:

```
vocab_size_target=len(tknizer_target.word_index.keys())
print(vocab_size_target)
vocab_size_source=len(tknizer_source.word_index.keys())
print(vocab_size_source)
```

```
92
103
```

In [28]:

```
tknizer_target.word_index['\t'], tknizer_target.word_index['\n']
```

Out[28]:

```
(20, 85)
```

In [29]:

```
class Encoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns output sequence
    '''

    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):

        #Initialize Embedding layer
        #Intialize Encoder LSTM layer
        super().__init__()
        self.vocab_size = inp_vocab_size
        self.embedding_size = embedding_size
        self.input_length = input_length
        self.lstm_size= lstm_size
        self.lstm_output=0
        self.embedding = tf.keras.layers.Embedding(input_dim=self.vocab_size, output_dim=self.embedding_s
                           mask_zero=True, name="embedding_layer_encoder")
        self.lstm = tf.keras.layers.LSTM(self.lstm_size, return_state=True, return_sequences=True, name="


    def call(self,input_sequence,states):

      '''
          This function takes a sequence input and the initial states of the encoder.
          Pass the input_sequence input to the Embedding layer, Pass the embedding layer oupulstm_state_h
          returns -- All encoder_outputs, last time steps hidden and cell state
      '''

      input_embedd                            = self.embedding(input_sequence)
      lstm_state_h,lstm_state_c= states[0],states[1]
      self.lstm_output,lstm_state_h,lstm_state_c=self.lstm(input_embedd)
      return self.lstm_output,lstm_state_h,lstm_state_c



    def initialize_states(self,batch_size):
      '''
      Given a batch size it will return intial hidden state and intial cell state.
      If batch size is 32- Hidden state is zeros of size [32,lstm_units], cell state zeros is of size [32
      '''
      return [tf.zeros((batch_size,self.lstm_size)),tf.zeros((batch_size,self.lstm_size))]
```

In [30]:

```
class Attention(tf.keras.layers.Layer):
  '''
    Class the calculates score based on the scoring_function using Bahdanu attention mechanism.
  '''
  def __init__(self,scoring_function, att_units):
    super().__init__()
    self.scoring_function=scoring_function
    # Please go through the reference notebook and research paper to complete the scoring functions
```

```
    if self.scoring_function=='dot':
      # Intialize variables needed for Dot score function here
      pass
    if self.scoring_function == 'general':
      # Intialize variables needed for General score function here
      self.weight=tf.keras.layers.Dense(att_units)
    elif self.scoring_function == 'concat':
      # Intialize variables needed for Concat score function here
      self.weight1=tf.keras.layers.Dense(att_units)
      self.weight2=tf.keras.layers.Dense(att_units)
      self.v=tf.keras.layers.Dense(1)


  def call(self,decoder_hidden_state,encoder_output):
    '''
      Attention mechanism takes two inputs current step -- decoder_hidden_state and all the encoder_outpu
      * Based on the scoring function we will find the score or similarity between decoder_hidden_state a
        Multiply the score function with your encoder_outputs to get the context vector.
        Function returns context vector and attention weights(softmax - scores)
    '''

    if self.scoring_function == 'dot':
        # Implement Dot score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=2)
        value=tf.matmul(encoder_output,decoder_hidden_state)
    elif self.scoring_function == 'general':
        # Implement General score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=2)
        value=tf.matmul(self.weight(encoder_output),decoder_hidden_state)
    elif self.scoring_function == 'concat':
        # Implement General score function here
        decoder_hidden_state=tf.expand_dims(decoder_hidden_state,axis=1)
        value=self.v(tf.nn.tanh(self.weight1(decoder_hidden_state)+self.weight2(encoder_output)))
    attention_weights=tf.nn.softmax(value,axis=1)
    context_vector=attention_weights*encoder_output
    return tf.reduce_sum(context_vector,axis=1),attention_weights
```

In [31]:

```
class One_Step_Decoder(tf.keras.Model):
  def __init__(self,tar_vocab_size, embedding_dim, input_length, dec_units ,score_fun ,att_units):

    # Initialize decoder embedding layer, LSTM and any other objects needed
    super().__init__()
    self.tar_vocab_size = tar_vocab_size
    self.embedding_dim = embedding_dim
    self.input_length = input_length
    self.dec_units = dec_units
    self.score_fun = score_fun
    self.att_units = att_units
    # we are using embedding_matrix and not training the embedding layer
    self.embedding = tf.keras.layers.Embedding(input_dim=self.tar_vocab_size, output_dim=self.embedding
                          mask_zero=True, name="embedding_layer_decoder", trainable=True)
    self.lstm = tf.keras.layers.LSTM(self.dec_units, return_sequences=True, return_state=True)
    self.dense = tf.keras.layers.Dense(self.tar_vocab_size)
    self.attention = Attention(self.score_fun,self.att_units)

  def call(self,input_to_decoder, encoder_output, state_h,state_c):
    '''
        One step decoder mechanisim step by step:
      A. Pass the input_to_decoder to the embedding layer and then get the output(batch_size,1,embedding_
      B. Using the encoder_output and decoder hidden state, compute the context vector.
      C. Concat the context vector with the step A output
      D. Pass the Step-C output to LSTM/GRU and get the decoder output and states(hidden and cell state)
      E. Pass the decoder output to dense layer(vocab size) and store the result into output.
      F. Return the states from step D, output from Step E, attention weights from Step -B
    '''
    output = self.embedding(input_to_decoder)
    context_vector,attention_weights = self.attention(state_h,encoder_output)
    context_vector1 = tf.expand_dims(context_vector,1)
    concat = tf.concat([output,context_vector1],axis=-1)
    decoder_output,state_h,state_c = self.lstm(concat,initial_state=[state_h,state_c])
    final_output = self.dense(decoder_output)
    final_output = tf.reshape(final_output,(-1,final_output.shape[2]))
    return final_output,state_h,state_c,attention_weights,context_vector
```

In [32]:

```
class Decoder(tf.keras.Model):
    def __init__(self,out_vocab_size, embedding_dim, input_length, dec_units ,score_fun ,att_units):
      #Intialize necessary variables and create an object from the class onestepdecoder
      super(Decoder,self).__init__()
      self.vocab_size = out_vocab_size
      self.embedding_dim = embedding_dim
      self.input_length = input_length
      self.dec_units=dec_units
      self.att_units=att_units
      self.score_fun=score_fun
      self.onestepdecoder=One_Step_Decoder(self.vocab_size,self.embedding_dim,self.input_length,self.dec_

    def call(self, input_to_decoder,encoder_output,decoder_hidden_state,decoder_cell_state ):

        #Initialize an empty Tensor array, that will store the outputs at each and every time step
        all_outputs=tf.TensorArray(tf.float32,size=tf.shape(input_to_decoder)[1])
        #Create a tensor array as shown in the reference notebook
        #Iterate till the length of the decoder input
        for timestep in range(tf.shape(input_to_decoder)[1]):
            # Call onestepdecoder for each token in decoder_input
            output,state_h,state_c,attention_weights,context_vector=self.onestepdecoder(input_to_decoder[

            # Store the output in tensorarray
            all_outputs=all_outputs.write(timestep,output)
        all_outputs=tf.transpose(all_outputs.stack(),[1,0,2])
        # Return the tensor array
        return all_outputs
```

In [44]:

```
class encoder_decoder(tf.keras.Model):
  def __init__(self,encoder_inputs_length,decoder_inputs_length, output_vocab_size,batch_size,score_fun):
    #Intialize objects from encoder decoder
    super().__init__() # https://stackoverflow.com/a/27134600/4084039
    self.batch_size=batch_size
    self.encoder = Encoder(vocab_size_source+1,300,100,encoder_inputs_length)
    self.decoder = Decoder(vocab_size_target+1,300,decoder_inputs_length,100,score_fun,100)

  def call(self,data):
    #Intialize encoder states, Pass the encoder_sequence to the embedding layer
    # Decoder initial states are encoder final states, Initialize it accordingly
    # Pass the decoder sequence,encoder_output,decoder states to Decoder
    # return the decoder output
    input,output = data[0], data[1]
    initial_state=self.encoder.initialize_states(self.batch_size)
    encoder_output, encoder_h, encoder_c = self.encoder(input,initial_state)
    decoder_output= self.decoder(output, encoder_output, encoder_h, encoder_c)
    return decoder_output
```

In [45]:

```
#https://www.tensorflow.org/tutorials/text/image_captioning#model
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')
def loss_function(real, pred):
    """ Custom loss function that will not consider the loss for padded zeros.
    why are we using this, can't we use simple sparse categorical crossentropy?
    Yes, you can use simple sparse categorical crossentropy as loss like we did in task-1. But in this lo
    for the padded zeros. i.e when the input is zero then we donot need to worry what the output is. This
    during preprocessing to make equal length for all the sentences.

    """


    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)

    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask

    return tf.reduce_mean(loss_)
```

In [46]:

```
class Dataset:
    def __init__(self, df, tknizer_source, tknizer_target, source_len,target_len):
```

```
        self.encoder_inps = df['source'].values
        self.decoder_inps = df['target_in'].values
        self.decoder_outs = df['target_out'].values
        self.tknizer_target = tknizer_target
        self.tknizer_source = tknizer_source
        self.source_len = source_len
        self.target_len = target_len


    def __getitem__(self, i):
        self.encoder_seq = self.tknizer_source.texts_to_sequences([self.encoder_inps[i]]) # need to pass
        self.decoder_inp_seq = self.tknizer_target.texts_to_sequences([self.decoder_inps[i]])
        self.decoder_out_seq = self.tknizer_target.texts_to_sequences([self.decoder_outs[i]])

        self.encoder_seq = pad_sequences(self.encoder_seq, maxlen=self.source_len, dtype='int32', padding
        self.decoder_inp_seq = pad_sequences(self.decoder_inp_seq, maxlen=self.target_len, dtype='int32',
        self.decoder_out_seq = pad_sequences(self.decoder_out_seq, maxlen=self.target_len, dtype='int32',
        return self.encoder_seq, self.decoder_inp_seq, self.decoder_out_seq

    def __len__(self): # your model.fit_gen requires this function
        return len(self.encoder_inps)


class Dataloder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size=1):
        self.dataset = dataset
        self.batch_size = batch_size
        self.indexes = np.arange(len(self.dataset.encoder_inps))


    def __getitem__(self, i):
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.squeeze(np.stack(samples, axis=1), axis=0) for samples in zip(*data)]
        # we are creating data like ([italian, english_inp], english_out) these are already converted in
        return tuple([[batch[0],batch[1]],batch[2]])

    def __len__(self):  # your model.fit_gen requires this function
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        self.indexes = np.random.permutation(self.indexes)
```

In [47]:

```
train_dataset = Dataset(train, tknizer_source, tknizer_target,170,200)
test_dataset  = Dataset(validation, tknizer_source, tknizer_target,170,200)

train_dataloader = Dataloder(train_dataset, batch_size=512)
test_dataloader = Dataloder(test_dataset, batch_size=20)


print(train_dataloader[0][0][0].shape, train_dataloader[0][0][1].shape, train_dataloader[0][1].shape)
```

```
(512, 170) (512, 200) (512, 200)
```

In [48]:

```
#Create an object of encoder_decoder Model class,
# Compile the model and fit the model
# Implement teacher forcing while training your model. You can do it two ways.
# Prepare your data, encoder_input,decoder_input and decoder_output
# if decoder input is
# <start> Hi how are you
# decoder output should be
# Hi How are you <end>
# i.e when you have send <start>-- decoder predicted Hi, 'Hi' decoder predicted 'How' .. e.t.c

# or

# model.fit([train_ita,train_eng],train_eng[:,1:]..)
# Note: If you follow this approach some grader functions might return false and this is fine.
model  = encoder_decoder(encoder_inputs_length=170,decoder_inputs_length=200,output_vocab_size=vocab_size
optimizer = tf.keras.optimizers.Adam(0.01)
model.compile(optimizer=optimizer,loss=loss_function)
train_steps=train.shape[0]//512
```

```
valid_steps=validation.shape[0]//20
model.fit_generator(train_dataloader, steps_per_epoch=train_steps, epochs=100, validation_data=test_datal
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1940: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/100
3/3 [==============================] - 16s 4s/step - loss: 1.4901 - val_loss: 0.9755
Epoch 2/100
3/3 [==============================] - 9s 3s/step - loss: 1.1887 - val_loss: 0.9433
Epoch 3/100
3/3 [==============================] - 9s 3s/step - loss: 1.1406 - val_loss: 0.9031
Epoch 4/100
3/3 [==============================] - 9s 3s/step - loss: 1.0853 - val_loss: 0.8608
Epoch 5/100
3/3 [==============================] - 8s 3s/step - loss: 1.0354 - val_loss: 0.8301
Epoch 6/100
3/3 [==============================] - 9s 3s/step - loss: 0.9966 - val_loss: 0.8033
Epoch 7/100
3/3 [==============================] - 9s 3s/step - loss: 0.9663 - val_loss: 0.7835
Epoch 8/100
3/3 [==============================] - 9s 3s/step - loss: 0.9445 - val_loss: 0.7679
Epoch 9/100
3/3 [==============================] - 9s 3s/step - loss: 0.9295 - val_loss: 0.7558
Epoch 10/100
3/3 [==============================] - 9s 3s/step - loss: 0.9178 - val_loss: 0.7471
Epoch 11/100
3/3 [==============================] - 9s 3s/step - loss: 0.9097 - val_loss: 0.7412
Epoch 12/100
3/3 [==============================] - 9s 3s/step - loss: 0.9030 - val_loss: 0.7366
Epoch 13/100
3/3 [==============================] - 9s 3s/step - loss: 0.8976 - val_loss: 0.7325
Epoch 14/100
3/3 [==============================] - 9s 3s/step - loss: 0.8939 - val_loss: 0.7282
Epoch 15/100
3/3 [==============================] - 9s 3s/step - loss: 0.8906 - val_loss: 0.7239
Epoch 16/100
3/3 [==============================] - 9s 3s/step - loss: 0.8881 - val_loss: 0.7207
Epoch 17/100
3/3 [==============================] - 9s 3s/step - loss: 0.8860 - val_loss: 0.7189
Epoch 18/100
3/3 [==============================] - 9s 3s/step - loss: 0.8842 - val_loss: 0.7174
Epoch 19/100
3/3 [==============================] - 9s 3s/step - loss: 0.8826 - val_loss: 0.7158
Epoch 20/100
3/3 [==============================] - 9s 3s/step - loss: 0.8811 - val_loss: 0.7151
Epoch 21/100
3/3 [==============================] - 9s 3s/step - loss: 0.8800 - val_loss: 0.7137
Epoch 22/100
3/3 [==============================] - 9s 3s/step - loss: 0.8787 - val_loss: 0.7135
Epoch 23/100
3/3 [==============================] - 9s 3s/step - loss: 0.8778 - val_loss: 0.7124
Epoch 24/100
3/3 [==============================] - 9s 3s/step - loss: 0.8767 - val_loss: 0.7114
Epoch 25/100
3/3 [==============================] - 9s 3s/step - loss: 0.8758 - val_loss: 0.7103
Epoch 26/100
3/3 [==============================] - 9s 3s/step - loss: 0.8748 - val_loss: 0.7095
Epoch 27/100
3/3 [==============================] - 9s 3s/step - loss: 0.8737 - val_loss: 0.7091
Epoch 28/100
3/3 [==============================] - 9s 3s/step - loss: 0.8729 - val_loss: 0.7087
Epoch 29/100
3/3 [==============================] - 9s 3s/step - loss: 0.8719 - val_loss: 0.7079
Epoch 30/100
3/3 [==============================] - 9s 3s/step - loss: 0.8708 - val_loss: 0.7069
Epoch 31/100
3/3 [==============================] - 9s 3s/step - loss: 0.8701 - val_loss: 0.7063
Epoch 32/100
3/3 [==============================] - 9s 3s/step - loss: 0.8694 - val_loss: 0.7051
Epoch 33/100
3/3 [==============================] - 9s 3s/step - loss: 0.8683 - val_loss: 0.7045
Epoch 34/100
3/3 [==============================] - 9s 3s/step - loss: 0.8678 - val_loss: 0.7056
Epoch 35/100
3/3 [==============================] - 9s 3s/step - loss: 0.8677 - val_loss: 0.7040
Epoch 36/100
```

```
Epoch 36/100
3/3 [==============================] - 9s 3s/step - loss: 0.8669 - val_loss: 0.7039
Epoch 37/100
3/3 [==============================] - 9s 3s/step - loss: 0.8657 - val_loss: 0.7024
Epoch 38/100
3/3 [==============================] - 9s 3s/step - loss: 0.8650 - val_loss: 0.7021
Epoch 39/100
3/3 [==============================] - 9s 3s/step - loss: 0.8641 - val_loss: 0.7024
Epoch 40/100
3/3 [==============================] - 9s 3s/step - loss: 0.8634 - val_loss: 0.7009
Epoch 41/100
3/3 [==============================] - 9s 3s/step - loss: 0.8625 - val_loss: 0.6999
Epoch 42/100
3/3 [==============================] - 9s 3s/step - loss: 0.8614 - val_loss: 0.6996
Epoch 43/100
3/3 [==============================] - 9s 3s/step - loss: 0.8608 - val_loss: 0.7001
Epoch 44/100
3/3 [==============================] - 9s 3s/step - loss: 0.8603 - val_loss: 0.6991
Epoch 45/100
3/3 [==============================] - 9s 3s/step - loss: 0.8590 - val_loss: 0.6976
Epoch 46/100
3/3 [==============================] - 9s 3s/step - loss: 0.8581 - val_loss: 0.6970
Epoch 47/100
3/3 [==============================] - 9s 3s/step - loss: 0.8571 - val_loss: 0.6978
Epoch 48/100
3/3 [==============================] - 9s 3s/step - loss: 0.8567 - val_loss: 0.6974
Epoch 49/100
3/3 [==============================] - 9s 3s/step - loss: 0.8557 - val_loss: 0.6963
Epoch 50/100
3/3 [==============================] - 9s 3s/step - loss: 0.8547 - val_loss: 0.6966
Epoch 51/100
3/3 [==============================] - 9s 3s/step - loss: 0.8538 - val_loss: 0.6948
Epoch 52/100
3/3 [==============================] - 9s 3s/step - loss: 0.8528 - val_loss: 0.6954
Epoch 53/100
3/3 [==============================] - 9s 3s/step - loss: 0.8519 - val_loss: 0.6943
Epoch 54/100
3/3 [==============================] - 9s 3s/step - loss: 0.8514 - val_loss: 0.6932
Epoch 55/100
3/3 [==============================] - 9s 3s/step - loss: 0.8500 - val_loss: 0.6941
Epoch 56/100
3/3 [==============================] - 9s 3s/step - loss: 0.8489 - val_loss: 0.6918
Epoch 57/100
3/3 [==============================] - 9s 3s/step - loss: 0.8479 - val_loss: 0.6905
Epoch 58/100
3/3 [==============================] - 9s 3s/step - loss: 0.8462 - val_loss: 0.6910
Epoch 59/100
3/3 [==============================] - 9s 3s/step - loss: 0.8456 - val_loss: 0.6909
Epoch 60/100
3/3 [==============================] - 9s 3s/step - loss: 0.8443 - val_loss: 0.6887
Epoch 61/100
3/3 [==============================] - 9s 3s/step - loss: 0.8432 - val_loss: 0.6889
Epoch 62/100
3/3 [==============================] - 9s 3s/step - loss: 0.8418 - val_loss: 0.6898
Epoch 63/100
3/3 [==============================] - 9s 3s/step - loss: 0.8406 - val_loss: 0.6883
Epoch 64/100
3/3 [==============================] - 9s 3s/step - loss: 0.8395 - val_loss: 0.6883
Epoch 65/100
3/3 [==============================] - 9s 3s/step - loss: 0.8382 - val_loss: 0.6871
Epoch 66/100
3/3 [==============================] - 9s 3s/step - loss: 0.8366 - val_loss: 0.6870
Epoch 67/100
3/3 [==============================] - 9s 3s/step - loss: 0.8359 - val_loss: 0.6870
Epoch 68/100
3/3 [==============================] - 9s 3s/step - loss: 0.8355 - val_loss: 0.6862
Epoch 69/100
3/3 [==============================] - 9s 3s/step - loss: 0.8339 - val_loss: 0.6865
Epoch 70/100
3/3 [==============================] - 9s 3s/step - loss: 0.8329 - val_loss: 0.6881
Epoch 71/100
3/3 [==============================] - 9s 3s/step - loss: 0.8314 - val_loss: 0.6845
Epoch 72/100
3/3 [==============================] - 9s 3s/step - loss: 0.8301 - val_loss: 0.6841
Epoch 73/100
3/3 [==============================] - 9s 3s/step - loss: 0.8285 - val_loss: 0.6841
Epoch 74/100
```

```
3/3 [==============================] - 9s 3s/step - loss: 0.8278 - val_loss: 0.6835
Epoch 75/100
3/3 [==============================] - 9s 3s/step - loss: 0.8263 - val_loss: 0.6844
Epoch 76/100
3/3 [==============================] - 9s 3s/step - loss: 0.8247 - val_loss: 0.6831
Epoch 77/100
3/3 [==============================] - 9s 3s/step - loss: 0.8237 - val_loss: 0.6829
Epoch 78/100
3/3 [==============================] - 9s 3s/step - loss: 0.8227 - val_loss: 0.6828
Epoch 79/100
3/3 [==============================] - 9s 3s/step - loss: 0.8223 - val_loss: 0.6830
Epoch 80/100
3/3 [==============================] - 9s 3s/step - loss: 0.8211 - val_loss: 0.6822
Epoch 81/100
3/3 [==============================] - 9s 3s/step - loss: 0.8194 - val_loss: 0.6826
Epoch 82/100
3/3 [==============================] - 9s 3s/step - loss: 0.8181 - val_loss: 0.6818
Epoch 83/100
3/3 [==============================] - 9s 3s/step - loss: 0.8166 - val_loss: 0.6826
Epoch 84/100
3/3 [==============================] - 9s 3s/step - loss: 0.8159 - val_loss: 0.6815
Epoch 85/100
3/3 [==============================] - 9s 3s/step - loss: 0.8145 - val_loss: 0.6816
Epoch 86/100
3/3 [==============================] - 9s 3s/step - loss: 0.8129 - val_loss: 0.6804
Epoch 87/100
3/3 [==============================] - 9s 3s/step - loss: 0.8125 - val_loss: 0.6810
Epoch 88/100
3/3 [==============================] - 9s 3s/step - loss: 0.8112 - val_loss: 0.6810
Epoch 89/100
3/3 [==============================] - 9s 3s/step - loss: 0.8110 - val_loss: 0.6800
Epoch 90/100
3/3 [==============================] - 9s 3s/step - loss: 0.8086 - val_loss: 0.6817
Epoch 91/100
3/3 [==============================] - 9s 3s/step - loss: 0.8079 - val_loss: 0.6784
Epoch 92/100
3/3 [==============================] - 9s 3s/step - loss: 0.8066 - val_loss: 0.6810
Epoch 93/100
3/3 [==============================] - 9s 3s/step - loss: 0.8053 - val_loss: 0.6817
Epoch 94/100
3/3 [==============================] - 9s 3s/step - loss: 0.8047 - val_loss: 0.6798
Epoch 95/100
3/3 [==============================] - 9s 3s/step - loss: 0.8034 - val_loss: 0.6792
Epoch 96/100
3/3 [==============================] - 9s 3s/step - loss: 0.8022 - val_loss: 0.6800
Epoch 97/100
3/3 [==============================] - 9s 3s/step - loss: 0.8015 - val_loss: 0.6783
Epoch 98/100
3/3 [==============================] - 9s 3s/step - loss: 0.7999 - val_loss: 0.6782
Epoch 99/100
3/3 [==============================] - 9s 3s/step - loss: 0.7985 - val_loss: 0.6777
Epoch 100/100
3/3 [==============================] - 9s 3s/step - loss: 0.7968 - val_loss: 0.6786
```

Out[48]:

```
<tensorflow.python.keras.callbacks.History at 0x7f17e1ef9450>
```

In [38]:

```
model.save_weights("model7.h5")
batch_size=512
units=100
```

In [39]:

```
def predict(input_sentence):

  '''
  A. Given input sentence, convert the sentence into integers using tokenizer used earlier
  B. Pass the input_sequence to encoder. we get encoder_outputs, last time step hidden and cell state
  C. Initialize index of <start> as input to decoder. and encoder final states as input_states to onestep
  D. till we reach max_length of decoder or till the model predicted word <end>:
        predictions, input_states, attention_weights = model.layers[1].onestepdecoder(input_to_decoder,
        Save the attention weights
        And get the word using the tokenizer(word index) and then store it in a string.
  E. Call plot_attention(#params)
  F. Return the predicted sentence
  '''
  initial_state_enc=[np.zeros((batch_size,units)),np.zeros((batch_size,units))]
  inp_seq = tknizer_source.texts_to_sequences([input_sentence])
  inp_seq = pad_sequences(inp_seq,padding='post',maxlen=170)
```

```
  en_outputs,state_h , state_c = model.layers[0](tf.constant(inp_seq),initial_state_enc)
  cur_vec = tf.constant([[tknizer_target.word_index['\t']]])
  pred = []
  #Here 20 is the max_length of the sequence
  for i in range(200):
    output,state_h,state_c,attention_weights,context_vector = model.layers[1].onestepdecoder(cur_vec,en_o
    cur_vec = np.reshape(np.argmax(output), (1, 1))
    pred.append(tknizer_target.index_word[cur_vec[0][0]])
    if(pred[-1]=='\n'):
      break
    translated_sentence = ''.join(pred)
  return translated_sentence
```

In [40]:

```
for i in validation['source']:
  pred=predict(i)
  print(i)
  print(pred)
```

```
Hello h r u
Youcht t t t yot
thathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathatha
athathathathathathathathathathathathathathathathathathathathathathatha
We saw fiona xie at taka... Haha...
Yout t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
if u are in town. I take taxi myself.
His at athat t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
U free tmr? Wana watch finding nemo...
Whathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathathatha
athathathathathathathathathathathathathathathathathathathathathathathathathathathathathath
Ü c da glasses nice n cn sit properly on è nose... It feels comfy then buy lor... Wat haf ü bot oredi?
I at at t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
HAI!Nice 2 meet u! :- >im gal 18 chinese. U?hmm... <: > my hp num is 0165460953 more cheaper sms by
phone!
His.
30265306306:300163090963096300163090963096300163090963096300163090963096300163090963096300163090963096300
0963096300163090963096300163090963096300163090963096300163090963096300163090963096300
Haha. Sure anot? Oni a bit. Haha. Tats bc u go thailand mahz. Mine not more thn 15.
Houthathath t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Ok...I wait 4 u outside toilet...Raffles city rite?
Hathathath t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Ok then when i reach i go collect ticket first. Now i nothing to do watch tv.
I t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Call me
Hi, bou t t t t yot t t t th yot t t t t th yot t t t t th yot t t t t th yot t t t t th yot t t t t th
yot t t t t th yot t t t t th yot t t t t th yot t t t t th yot t t t t th yot t t t t th yot t
Wow rot at home.... Ha ha..... Raining ma good weather to sleep.... Ha ha.....
Whathatht t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Get smth lo...Den nvm ba...Heh.
I t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
i think what i hav told so far is no big secret...
I t it it t it t it t t t t t t t t t t t t t t t t t t t t t t t t it t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Ay i exam period ah...seow ah. Dyin--
His ay ay ay t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Help me record e guess3 ok? E back part...
Hoout t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
how u noe there's no better ans ..
Houthathath t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
It's ok.. Happy studying..
Hing ay t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
we just come out from turf club. both of is made $800 each.
I t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Just left ofc...
Yoffffffff t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
Tomw dinner at 7.30 either jap food or ron suggest sizzler. Wat u prefer?
SMay at t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t t
```

In [41]:

```
# Predict on 1000 random sentences on test data and calculate the average BLEU score of these sentences.
import nltk.translate.bleu_score as bleu
bleu_scores_lst=[]
for i in validation[:]['source']:
  reference = [i.split(),] # the original
  predicted=predict(i)
  translation = predicted.split()
  values=bleu.sentence_bleu(reference, translation)
  bleu_scores_lst.append(values)

# https://www.nltk.org/_modules/nltk/translate/bleu_score.html
```

```
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
```

In [42]:

```
average_bleu_scores=sum(bleu_scores_lst)/len(bleu_scores_lst)
print("Average BLEU score of these 20 test data sentences is: ",average_bleu_scores)
```

```
Average BLEU score of these 20 test data sentences is:  0.015891448522335924
```

In [43]:

```
bleu_scores_lst
```

Out[43]:

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.3178289704467185]
```

In [ ]: