In [1]:

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

```
Downloading...
From: https://drive.google.com/uc?id=1OurDQUtbWQacvT32HMqFL7vIUrSMllOp
To: /content/preprocessed_data.csv
100% 300k/300k [00:00<00:00, 9.69MB/s]
```

In [8]:

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2
.8.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.41.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.5.30
)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (5.
0.2)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests
->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kag
gle) (2.10)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python
-slugify->kaggle) (1.3)
```

In [9]:

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 /root/.kaggle/kaggle.json
!kaggle datasets download -d yekenot/fasttext-crawl-300d-2m
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
Downloading fasttext-crawl-300d-2m.zip to /content
 99% 1.43G/1.44G [00:06<00:00, 234MB/s]
100% 1.44G/1.44G [00:06<00:00, 223MB/s]
```

In [10]:

```
!7z e fasttext-crawl-300d-2m.zip -o/content -r
```

```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Xeon(R) CPU @
2.20GHz (406F0),ASM,AES-NI)

Scanning the drive for archives:
  0M Scan                           1 file, 1545551987 bytes (1474 MiB)

Extracting archive: fasttext-crawl-300d-2m.zip
--
Path = fasttext-crawl-300d-2m.zip
Type = zip
Physical Size = 1545551987

  0%              0% - crawl-300d-2M.vec
                           1% - crawl-300d-2M.vec
                  2% - crawl-300d-2M.vec
                           3% - crawl-300d-2M.vec
                  4% - crawl-300d-2M.vec
                           5% - crawl-300d-2M.vec
                  6% - crawl-300d-2M.vec
                           7% - crawl-300d-2M.vec
                  8% - crawl-300d-2M.vec
                           9% - crawl-300d-2M.vec
                 10% - crawl-300d-2M.vec
                          11% - crawl-300d-2M.vec
                          12% - crawl-300d-2M.vec
                          13% - crawl-300d-2M.vec
                          14% - crawl-300d-2M.vec
                          15% - crawl-300d-2M.vec
                          16% - crawl-300d-2M.vec
                          17% - crawl-300d-2M.vec
                          18% - crawl-300d-2M.vec
                          19% - crawl-300d-2M.vec
                          20% - crawl-300d-2M.vec
                          21% - crawl-300d-2M.vec
                          22% - crawl-300d-2M.vec
```

23% - crawl-300d-2M.vec
24% - crawl-300d-2M.vec
25% - crawl-300d-2M.vec
26% - crawl-300d-2M.vec
27% - crawl-300d-2M.vec
28% - crawl-300d-2M.vec
29% - crawl-300d-2M.vec
30% - crawl-300d-2M.vec
31% - crawl-300d-2M.vec
32% - crawl-300d-2M.vec
33% - crawl-300d-2M.vec
34% - crawl-300d-2M.vec
35% - crawl-300d-2M.vec
36% - crawl-300d-2M.vec
37% - crawl-300d-2M.vec
38% - crawl-300d-2M.vec
39% - crawl-300d-2M.vec
40% - crawl-300d-2M.vec
41% - crawl-300d-2M.vec
42% - crawl-300d-2M.vec
43% - crawl-300d-2M.vec
44% - crawl-300d-2M.vec
45% - crawl-300d-2M.vec
46% - crawl-300d-2M.vec
47% - crawl-300d-2M.vec
48% - crawl-300d-2M.vec
49% - crawl-300d-2M.vec
50% - crawl-300d-2M.vec
51% - crawl-300d-2M.vec
52% - crawl-300d-2M.vec
53% - crawl-300d-2M.vec
54% - crawl-300d-2M.vec
55% - crawl-300d-2M.vec
56% - crawl-300d-2M.vec
57% - crawl-300d-2M.vec
58% - crawl-300d-2M.vec
59% - crawl-300d-2M.vec
60% - crawl-300d-2M.vec
61% - crawl-300d-2M.vec
62% - crawl-300d-2M.vec
63% - crawl-300d-2M.vec
64% - crawl-300d-2M.vec
65% - crawl-300d-2M.vec
66% - crawl-300d-2M.vec
67% - crawl-300d-2M.vec
68% - crawl-300d-2M.vec
69% - crawl-300d-2M.vec
70% - crawl-300d-2M.vec
71% - crawl-300d-2M.vec
72% - crawl-300d-2M.vec
73% - crawl-300d-2M.vec
74% - crawl-300d-2M.vec
75% - crawl-300d-2M.vec
76% - crawl-300d-2M.vec
77% - crawl-300d-2M.vec
78% - crawl-300d-2M.vec
79% - crawl-300d-2M.vec
80% - crawl-300d-2M.vec
81% - crawl-300d-2M.vec
82% - crawl-300d-2M.vec
83% - crawl-300d-2M.vec
84% - crawl-300d-2M.vec
85% - crawl-300d-2M.vec
86% - crawl-300d-2M.vec
87% - crawl-300d-2M.vec
88% - crawl-300d-2M.vec
89% - crawl-300d-2M.vec
90% - crawl-300d-2M.vec
91% - crawl-300d-2M.vec
92% - crawl-300d-2M.vec
93% - crawl-300d-2M.vec
94% - crawl-300d-2M.vec
95% - crawl-300d-2M.vec
96% - crawl-300d-2M.vec
97% - crawl-300d-2M.vec
98% - crawl-300d-2M.vec
99% - crawl-300d-2M.vec

```
100% - crawl-300d-2M.vec
100% 1                  Everything is Ok

Size:       4516698366
Compressed: 1545551987
```

In [11]:

```
#Importing necessary libraries
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 [12]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def fasttextModel(gloveFile):
    print ("Loading Fasttext Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}#for storing word and the corresponding embedding vector for that word
    for line in f:
        splitLine = line.split()#splitting the line and storing it in a list
        word = splitLine[0]#getting the first element and storing it in word
        embedding = np.array([float(val) for val in splitLine[1:]])#obtaining corresponding vector for th
        model[word] = embedding#storing word as key and embedding vector for that word as value
    print ("Done.",len(model)," words loaded!")
    return model
model = fasttextModel('/content/crawl-300d-2M.vec')
```

```
Loading Fasttext Model
Done. 2000000  words loaded!
```

In [13]:

```
df=pd.read_csv('preprocessed_data.csv')#creating DataFrame using preprocessed_data.csv
```

In [14]:

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

Out[14]:

	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 [15]:

```
def preprocess(x):#removing last character
  x=x[:-1]
  return x
```

In [16]:

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

In [17]:

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

Out[17]:

	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 [18]:

```
df.shape
```

Out[18]:

```
(2000, 2)
```

In [19]:

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

In [20]:

```
df=df[df['source'].apply(length)<170]#removing the datapoints where the source sentence length is greater
df=df[df['target'].apply(length)<200]#removing the datapoints where the source sentence length is greater
```

In [21]:

```
df.shape
```

Out[21]:

```
(1990, 2)
```

In [22]:

```
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[22]:

	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 [23]:

```
df=df.drop('target',axis=1)#removing the target column
```

In [24]:

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

Out[24]:

	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 [25]:

```
from sklearn.model_selection import train_test_split
train, validation = train_test_split(df, test_size=0.01)#splitting the data in ratio 99:1
```

In [26]:

```
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 [27]:

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

In [28]:

```
vocab_size_target=len(tknizer_target.word_index.keys())#target vocab size
print(vocab_size_target)
vocab_size_source=len(tknizer_source.word_index.keys())#source vocab size
print(vocab_size_source)
```

```
3040
3711
```

In [29]:

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

Out[29]:

```
(1, 1440)
```

In [30]:

```
encoder_embedding_matrix = np.zeros((vocab_size_source+1, 300))
for word, i in tknizer_source.word_index.items():
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        encoder_embedding_matrix[i] = embedding_vector
```

In [31]:

```
decoder_embedding_matrix = np.zeros((vocab_size_target+1, 300))
for word, i in tknizer_target.word_index.items():
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        decoder_embedding_matrix[i] = embedding_vector
```

In [70]:

```
class Encoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns encoder-outputs,encoder_final_state_h,encod
    '''

    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.lstm_state_h=0
        self.lstm_state_c=0
        self.embedding = tf.keras.layers.Embedding(input_dim=self.vocab_size, output_dim=self.embedding_s
                           mask_zero=True,name="embedding_layer_encoder",weights=[encoder_embedding_matr
        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 ouput to encode
          returns -- encoder_output, last time step's 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 [3
      '''
      return [np.zeros((batch_size,self.lstm_size)),np.zeros((batch_size,self.lstm_size))]
```

In [71]:

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

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

        #Initialize Embedding layer
        #Intialize Decoder LSTM layer
        super().__init__()
        self.out_vocab_size = out_vocab_size
```

```
        self.embedding_size = embedding_size
        self.lstm_size = lstm_size
        self.input_length = input_length
        # we are using embedding_matrix and not training the embedding layer
        self.embedding = tf.keras.layers.Embedding(input_dim=self.out_vocab_size, output_dim=self.embeddi
                           mask_zero=True, name="embedding_layer_decoder",weights=[decoder_embedding_mat
        self.gru = tf.keras.layers.GRU(self.lstm_size, return_sequences=True, return_state=True, name="En

    def call(self,input_sequence,initial_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 ouput to decode

          returns -- decoder_output,decoder_final_state_h,decoder_final_state_c
        '''
        target_embedd = self.embedding(input_sequence)
        decoder_output,decoder_final_state_h = self.gru(target_embedd, initial_state=[initial_states[0]])
        return decoder_output,decoder_final_state_h
```

In [80]:

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

    def __init__(self,encoder_inputs_length,decoder_inputs_length, output_vocab_size,batch_size):

        #Create encoder object
        #Create decoder object
        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        super().__init__() # https://stackoverflow.com/a/27134600/4084039
        self.batch_size=batch_size
        self.encoder = Encoder(vocab_size_source+1,300,256,encoder_inputs_length)
        self.decoder = Decoder(vocab_size_target+1,300,256,decoder_inputs_length)
        self.dense   = tf.keras.layers.Dense(output_vocab_size, activation='softmax')


    def call(self,data):
        '''
        A. Pass the input sequence to Encoder layer -- Return encoder_output,encoder_final_state_h,encod
        B. Pass the target sequence to Decoder layer with intial states as encoder_final_state_h,encoder_
        C. Pass the decoder_outputs into Dense layer

        Return decoder_outputs
        '''
        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,decoder_final_state_h= self.decoder(output,[encoder_h])
        output                              = self.dense(decoder_output)
        return output
```

In [81]:

```
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',
        self.one_hot_encoded = np.zeros((len(self.decoder_out_seq),self.target_len,vocab_size_target),dty

        for i,sentence in enumerate(self.decoder_out_seq):
          for j,word in enumerate(sentence):
            self.one_hot_encoded[i,j,word]=1
```

```
        return self.encoder_seq, self.decoder_inp_seq, self.one_hot_encoded

    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 [82]:

```
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, 3040)
```

In [83]:

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

In [84]:

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

In [85]:

```
def changeLearningRate(epoch,lr):
    if epoch % 3 == 0:
      return lr*(0.5)
    return lr
```

In [86]:

```
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint,LearningRateScheduler
import datetime

early_stop = EarlyStopping(monitor='val_loss', patience=4, verbose=1)
check_point = ModelCheckpoint('best_model_1.h5', monitor='val_loss',  verbose=1, save_best_only=True, mod
lrschedule = LearningRateScheduler(changeLearningRate, verbose=1)
```

In [87]:

```
#Create an object of encoder_decoder Model class,
# Compile the model and fit the model
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='categorical_crossentropy',metrics=['accuracy'])
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
model.summary()
```

```
/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

Epoch 00001: LearningRateScheduler reducing learning rate to 0.0049999999888241291.
3/3 [==============================] - 1s 422ms/step - loss: 2.7390 - accuracy: 0.0507 - val_loss:
1.6437 - val_accuracy: 0.0946
Epoch 2/100

Epoch 00002: LearningRateScheduler reducing learning rate to 0.0049999999888241291.
3/3 [==============================] - 1s 399ms/step - loss: 2.1927 - accuracy: 0.0677 - val_loss:
1.5412 - val_accuracy: 0.0946
Epoch 3/100

Epoch 00003: LearningRateScheduler reducing learning rate to 0.0049999999888241291.
3/3 [==============================] - 1s 406ms/step - loss: 2.1509 - accuracy: 0.0669 - val_loss:
1.5111 - val_accuracy: 0.0901
Epoch 4/100

Epoch 00004: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 1s 412ms/step - loss: 2.0908 - accuracy: 0.0870 - val_loss:
1.5020 - val_accuracy: 0.1081
Epoch 5/100

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 1s 404ms/step - loss: 2.0622 - accuracy: 0.0852 - val_loss:
1.4941 - val_accuracy: 0.1126
Epoch 6/100

Epoch 00006: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 1s 406ms/step - loss: 2.0396 - accuracy: 0.0860 - val_loss:
1.4884 - val_accuracy: 0.1216
Epoch 7/100

Epoch 00007: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 1s 413ms/step - loss: 2.0213 - accuracy: 0.0895 - val_loss:
1.4831 - val_accuracy: 0.1126
Epoch 8/100

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 1s 419ms/step - loss: 2.0094 - accuracy: 0.0920 - val_loss:
1.4778 - val_accuracy: 0.1081
Epoch 9/100

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 1s 428ms/step - loss: 2.0009 - accuracy: 0.0930 - val_loss:
1.4739 - val_accuracy: 0.1081
Epoch 10/100

Epoch 00010: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 1s 409ms/step - loss: 1.9907 - accuracy: 0.0948 - val_loss:
1.4715 - val_accuracy: 0.1081
Epoch 11/100

Epoch 00011: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 1s 422ms/step - loss: 1.9860 - accuracy: 0.0952 - val_loss:
1.4690 - val_accuracy: 0.1126
Epoch 12/100

Epoch 00012: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 1s 412ms/step - loss: 1.9806 - accuracy: 0.0956 - val_loss:
1.4663 - val_accuracy: 0.1216
Epoch 13/100

Epoch 00013: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 1s 416ms/step - loss: 1.9759 - accuracy: 0.0973 - val_loss:
1.4653 - val_accuracy: 0.1216
Epoch 14/100

Epoch 00014: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 1s 416ms/step - loss: 1.9734 - accuracy: 0.0984 - val_loss:
1.4644 - val_accuracy: 0.1171
Epoch 15/100
```

```
Epoch 00015: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 1s 410ms/step - loss: 1.9708 - accuracy: 0.0995 - val_loss:
1.4636 - val_accuracy: 0.1216
Epoch 16/100

Epoch 00016: LearningRateScheduler reducing learning rate to 0.0001562499965075403.
3/3 [==============================] - 1s 417ms/step - loss: 1.9684 - accuracy: 0.1004 - val_loss:
1.4632 - val_accuracy: 0.1261
Epoch 17/100

Epoch 00017: LearningRateScheduler reducing learning rate to 0.0001562499965075403.
3/3 [==============================] - 1s 407ms/step - loss: 1.9670 - accuracy: 0.1004 - val_loss:
1.4627 - val_accuracy: 0.1261
Epoch 18/100

Epoch 00018: LearningRateScheduler reducing learning rate to 0.0001562499965075403.
3/3 [==============================] - 1s 414ms/step - loss: 1.9657 - accuracy: 0.1011 - val_loss:
1.4623 - val_accuracy: 0.1261
Epoch 19/100

Epoch 00019: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 1s 403ms/step - loss: 1.9645 - accuracy: 0.1015 - val_loss:
1.4621 - val_accuracy: 0.1261
Epoch 20/100

Epoch 00020: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 1s 410ms/step - loss: 1.9638 - accuracy: 0.1017 - val_loss:
1.4619 - val_accuracy: 0.1261
Epoch 21/100

Epoch 00021: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 1s 409ms/step - loss: 1.9632 - accuracy: 0.1015 - val_loss:
1.4617 - val_accuracy: 0.1306
Epoch 22/100

Epoch 00022: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 1s 413ms/step - loss: 1.9626 - accuracy: 0.1016 - val_loss:
1.4616 - val_accuracy: 0.1306
Epoch 23/100

Epoch 00023: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 1s 410ms/step - loss: 1.9622 - accuracy: 0.1015 - val_loss:
1.4615 - val_accuracy: 0.1306
Epoch 24/100

Epoch 00024: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 1s 414ms/step - loss: 1.9619 - accuracy: 0.1014 - val_loss:
1.4614 - val_accuracy: 0.1261
Epoch 25/100

Epoch 00025: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 1s 414ms/step - loss: 1.9616 - accuracy: 0.1016 - val_loss:
1.4614 - val_accuracy: 0.1261
Epoch 26/100

Epoch 00026: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 1s 411ms/step - loss: 1.9614 - accuracy: 0.1017 - val_loss:
1.4613 - val_accuracy: 0.1261
Epoch 27/100

Epoch 00027: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 1s 408ms/step - loss: 1.9613 - accuracy: 0.1018 - val_loss:
1.4613 - val_accuracy: 0.1261
Epoch 28/100

Epoch 00028: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
3/3 [==============================] - 1s 424ms/step - loss: 1.9611 - accuracy: 0.1020 - val_loss:
1.4612 - val_accuracy: 0.1261
Epoch 29/100

Epoch 00029: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
3/3 [==============================] - 1s 412ms/step - loss: 1.9610 - accuracy: 0.1020 - val_loss:
1.4612 - val_accuracy: 0.1261
Epoch 30/100

Epoch 00030: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
```

```
3/3 [==============================] - 1s 427ms/step - loss: 1.9609 - accuracy: 0.1019 - val_loss:
1.4612 - val_accuracy: 0.1261
Epoch 31/100

Epoch 00031: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 1s 424ms/step - loss: 1.9609 - accuracy: 0.1019 - val_loss:
1.4612 - val_accuracy: 0.1261
Epoch 32/100

Epoch 00032: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 1s 418ms/step - loss: 1.9608 - accuracy: 0.1020 - val_loss:
1.4612 - val_accuracy: 0.1261
Epoch 33/100

Epoch 00033: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 1s 425ms/step - loss: 1.9608 - accuracy: 0.1020 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 34/100

Epoch 00034: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 1s 416ms/step - loss: 1.9607 - accuracy: 0.1020 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 35/100

Epoch 00035: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 1s 420ms/step - loss: 1.9607 - accuracy: 0.1020 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 36/100

Epoch 00036: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 1s 423ms/step - loss: 1.9607 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 37/100

Epoch 00037: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 1s 418ms/step - loss: 1.9607 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 38/100

Epoch 00038: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 1s 410ms/step - loss: 1.9607 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 39/100

Epoch 00039: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 1s 421ms/step - loss: 1.9607 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 40/100

Epoch 00040: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 1s 420ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 41/100

Epoch 00041: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 1s 413ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 42/100

Epoch 00042: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 1s 412ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 43/100

Epoch 00043: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 1s 414ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 44/100

Epoch 00044: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 1s 421ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 45/100

Epoch 00045: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 1s 408ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
```

```
Epoch 46/100

Epoch 00046: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 1s 413ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 47/100

Epoch 00047: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 1s 415ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 48/100

Epoch 00048: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 1s 417ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 49/100

Epoch 00049: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
3/3 [==============================] - 1s 414ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 50/100

Epoch 00050: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
3/3 [==============================] - 1s 410ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 51/100

Epoch 00051: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
3/3 [==============================] - 1s 410ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 52/100

Epoch 00052: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 1s 416ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 53/100

Epoch 00053: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 1s 410ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 54/100

Epoch 00054: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 1s 417ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 55/100

Epoch 00055: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 1s 408ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 56/100

Epoch 00056: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 1s 409ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 57/100

Epoch 00057: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 1s 421ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 58/100

Epoch 00058: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 1s 411ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 59/100

Epoch 00059: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 1s 419ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 60/100

Epoch 00060: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 1s 416ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 61/100
```

```
Epoch 00061: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 1s 407ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 62/100

Epoch 00062: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 1s 398ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 63/100

Epoch 00063: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 1s 409ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 64/100

Epoch 00064: LearningRateScheduler reducing learning rate to 2.38418573772492e-09.
3/3 [==============================] - 1s 420ms/step - loss: 1.9606 - accuracy: 0.1019 - val_loss:
1.4611 - val_accuracy: 0.1261
Epoch 00064: early stopping
Model: "encoder_decoder_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_8 (Encoder)          multiple                  1683968
_________________________________________________________________
decoder_8 (Decoder)          multiple                  1340844
_________________________________________________________________
dense_8 (Dense)              multiple                  781280
=================================================================
Total params: 3,806,092
Trainable params: 1,780,192
Non-trainable params: 2,025,900
_________________________________________________________________
```

In [88]:

```
batch_size=512
units=256
```

In [92]:

```
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 decode
  D. till we reach max_length of decoder or till the model predicted word <end>:
        predicted_out,state_h,state_c=model.layers[1](dec_input,states)
        pass the predicted_out to the dense layer
        update the states=[state_h,state_c]
        And get the index of the wordcc with maximum probability of the dense layer output, using the t
        Update the input_to_decoder with current predictions
  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 = []
  #Here 43 is the max_length of the sequence
  for i in range(43):
    infe_output, state_h = model.layers[1](cur_vec,[state_h])
    infe_output = model.layers[2](infe_output)
    cur_vec = np.reshape(np.argmax(infe_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 [93]:

```
validation['target_in']
```

Out[93]:

```
899                         <start> Malays are all woods.
52      <start> Yeap. I will call you in a while? I'm ...
1597              <start> Where? Can I come and find you?
847     <start> No need for me to introduce. Someone a...
189     <start> Ben asks us to wait at the MRT bus sto...
1713                                          <start> Ok.
1818               <start> I'll be late. I will call you.
591                 <start> 2:30 then. Where will you be?
1450    <start> Haha. Hey, MERINA is my name. OK, fema...
175          <start> Joey: Hi! Does anybody want to chat?
1136                                  <start> No need.
786                     <start> Yes. Sure. Evening right?
1966                    <start> Oops. Sorry. I am coming.
1144    <start> Sigh, what's new man? So this is her n...
35      <start> I am working in NTUC Income, selling i...
1821    <start> Hey, are you in the LT already? I'm on...
1139    <start> You worse than me. Eat more of everyth...
1103    <start> Sigh. I still have my friends. Then ne...
1287                 <start> Don't know yet, see Andrew.
218     <start> No. Suddenly get sick one. Hehe. Your ...
Name: target_in, dtype: object
```

In [94]:

```
for i in validation['source']:
  predicted=predict(i)
  print("The predicted output is: ",predicted)
```

```
The predicted output is:  i don't know you you you you
The predicted output is:  hey i am you you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  hey i am you you you you you you you you
The predicted output is:  you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you
The predicted output is:  i don't know you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  hey i am you you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  hey i am you you you you you you you you
The predicted output is:  hey i am you you you you you you you you
The predicted output is:  i don't know you you you you you you you
The predicted output is:  i don't know you you you you you you you
```

In [95]:

```
# 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 [96]:

```
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:  0.059428793291298146
```

In [97]:

```
bleu_scores_lst
```

Out[97]:

```
[0,
 0.4578141331660858,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0.16842040746952797,
 0,
 0,
 0,
 0.5623413251903491,
 0]
```

Character_Level:

In [98]:

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

Out[98]:

	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 [99]:

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

In [100]:

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

In [101]:

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

Out[101]:

	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 [102]:

```
df.shape
```

Out[102]:

```
(2000, 2)
```

In [103]:

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

In [104]:

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

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

In [105]:

```
df.shape
```

Out[105]:

```
(1990, 2)
```

In [106]:

```
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[106]:

	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 [107]:

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

In [108]:

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

Out[108]:

	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 [109]:

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

In [110]:

```
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 [111]:

```
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 [112]:

```
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 [113]:

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

Out[113]:

```
(20, 85)
```

In [122]:

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

```
    '''
    Encoder model -- That takes a input sequence and returns encoder-outputs,encoder_final_state_h,encod
    '''

    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.lstm_state_h=0
        self.lstm_state_c=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 ouput to encode
          returns -- encoder_output, last time step's 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 [3
      '''
      return [np.zeros((batch_size,self.lstm_size)),np.zeros((batch_size,self.lstm_size))]
```

In [123]:

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

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

        #Initialize Embedding layer
        #Intialize Decoder LSTM layer
        super().__init__()
        self.out_vocab_size = out_vocab_size
        self.embedding_size = embedding_size
        self.lstm_size = lstm_size
        self.input_length = input_length
        # we are using embedding_matrix and not training the embedding layer
        self.embedding = tf.keras.layers.Embedding(input_dim=self.out_vocab_size, output_dim=self.embeddi
                           mask_zero=True, name="embedding_layer_decoder")
        self.gru = tf.keras.layers.GRU(self.lstm_size, return_sequences=True, return_state=True, name="En

    def call(self,input_sequence,initial_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 ouput to decode

          returns -- decoder_output,decoder_final_state_h,decoder_final_state_c
        '''
        target_embedd = self.embedding(input_sequence)
        decoder_output,decoder_final_state_h = self.gru(target_embedd, initial_state=[initial_states[0]])
        return decoder_output,decoder_final_state_h
```

In [136]:

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

```
    def __init__(self,encoder_inputs_length,decoder_inputs_length, output_vocab_size,batch_size):

        #Create encoder object
        #Create decoder object
        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        super().__init__() # https://stackoverflow.com/a/27134600/4084039
        self.batch_size=batch_size
        self.encoder = Encoder(vocab_size_source+1,512,256,encoder_inputs_length)
        self.decoder = Decoder(vocab_size_target+1,512,256,decoder_inputs_length)
        self.dense   = tf.keras.layers.Dense(output_vocab_size, activation='softmax')


    def call(self,data):
        '''
        A. Pass the input sequence to Encoder layer -- Return encoder_output,encoder_final_state_h,encod
        B. Pass the target sequence to Decoder layer with intial states as encoder_final_state_h,encoder_
        C. Pass the decoder_outputs into Dense layer

        Return decoder_outputs
        '''
        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,decoder_final_state_h= self.decoder(output,[encoder_h])
        output                              = self.dense(decoder_output)
        return output
```

In [137]:

```
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',
        self.one_hot_encoded = np.zeros((len(self.decoder_out_seq),self.target_len,vocab_size_target),dty

        for i,sentence in enumerate(self.decoder_out_seq):
          for j,word in enumerate(sentence):
            self.one_hot_encoded[i,j,word]=1




        return self.encoder_seq, self.decoder_inp_seq, self.one_hot_encoded

    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 [138]:

```
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, 92)
```

In [139]:

```
def changeLearningRate(epoch,lr):
    if epoch % 3 == 0:
      return lr*(0.5)
    return lr
```

In [140]:

```
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint,LearningRateScheduler
import datetime

early_stop = EarlyStopping(monitor='val_loss', patience=4, verbose=1)
check_point = ModelCheckpoint('best_model_1.h5', monitor='val_loss',  verbose=1, save_best_only=True, mod
lrschedule = LearningRateScheduler(changeLearningRate, verbose=1)
```

In [141]:

```
#Create an object of encoder_decoder Model class,
# Compile the model and fit the model
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='categorical_crossentropy',metrics=['accuracy'])
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
model.summary()
```

```
/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

Epoch 00001: LearningRateScheduler reducing learning rate to 0.004999999888241291.
3/3 [==============================] - 71s 35s/step - loss: 1.5354 - accuracy: 0.1540 - val_loss: 1.4060
- val_accuracy: 0.1758
Epoch 2/100

Epoch 00002: LearningRateScheduler reducing learning rate to 0.004999999888241291.
3/3 [==============================] - 2s 506ms/step - loss: 1.2939 - accuracy: 0.1434 - val_loss:
1.0846 - val_accuracy: 0.2136
Epoch 3/100

Epoch 00003: LearningRateScheduler reducing learning rate to 0.004999999888241291.
3/3 [==============================] - 2s 509ms/step - loss: 1.1312 - accuracy: 0.2131 - val_loss:
1.0058 - val_accuracy: 0.2218
Epoch 4/100

Epoch 00004: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 2s 512ms/step - loss: 1.0474 - accuracy: 0.2404 - val_loss:
0.9583 - val_accuracy: 0.2515
Epoch 5/100

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
```

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 2s 531ms/step - loss: 0.9980 - accuracy: 0.2758 - val_loss:
0.9234 - val_accuracy: 0.2804
Epoch 6/100

Epoch 00006: LearningRateScheduler reducing learning rate to 0.0024999999441206455.
3/3 [==============================] - 2s 505ms/step - loss: 0.9639 - accuracy: 0.2975 - val_loss:
0.8932 - val_accuracy: 0.2804
Epoch 7/100

Epoch 00007: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 2s 518ms/step - loss: 0.9391 - accuracy: 0.2971 - val_loss:
0.8745 - val_accuracy: 0.2967
Epoch 8/100

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 2s 506ms/step - loss: 0.9212 - accuracy: 0.3076 - val_loss:
0.8580 - val_accuracy: 0.2990
Epoch 9/100

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0012499999720603228.
3/3 [==============================] - 2s 512ms/step - loss: 0.9079 - accuracy: 0.3118 - val_loss:
0.8472 - val_accuracy: 0.3101
Epoch 10/100

Epoch 00010: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 2s 508ms/step - loss: 0.8997 - accuracy: 0.3154 - val_loss:
0.8415 - val_accuracy: 0.3101
Epoch 11/100

Epoch 00011: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 2s 516ms/step - loss: 0.8936 - accuracy: 0.3171 - val_loss:
0.8355 - val_accuracy: 0.3153
Epoch 12/100

Epoch 00012: LearningRateScheduler reducing learning rate to 0.0006249999860301614.
3/3 [==============================] - 2s 507ms/step - loss: 0.8876 - accuracy: 0.3226 - val_loss:
0.8304 - val_accuracy: 0.3153
Epoch 13/100

Epoch 00013: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 2s 502ms/step - loss: 0.8831 - accuracy: 0.3221 - val_loss:
0.8280 - val_accuracy: 0.3160
Epoch 14/100

Epoch 00014: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 2s 503ms/step - loss: 0.8804 - accuracy: 0.3223 - val_loss:
0.8257 - val_accuracy: 0.3123
Epoch 15/100

Epoch 00015: LearningRateScheduler reducing learning rate to 0.0003124999930150807.
3/3 [==============================] - 2s 513ms/step - loss: 0.8777 - accuracy: 0.3218 - val_loss:
0.8236 - val_accuracy: 0.3123
Epoch 16/100

Epoch 00016: LearningRateScheduler reducing learning rate to 0.00015624999650754035.
3/3 [==============================] - 2s 509ms/step - loss: 0.8756 - accuracy: 0.3221 - val_loss:
0.8225 - val_accuracy: 0.3131
Epoch 17/100

Epoch 00017: LearningRateScheduler reducing learning rate to 0.00015624999650754035.
3/3 [==============================] - 2s 512ms/step - loss: 0.8744 - accuracy: 0.3219 - val_loss:
0.8215 - val_accuracy: 0.3138
Epoch 18/100

Epoch 00018: LearningRateScheduler reducing learning rate to 0.00015624999650754035.
3/3 [==============================] - 2s 514ms/step - loss: 0.8732 - accuracy: 0.3213 - val_loss:
0.8206 - val_accuracy: 0.3093
Epoch 19/100

Epoch 00019: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 2s 507ms/step - loss: 0.8722 - accuracy: 0.3205 - val_loss:
0.8201 - val_accuracy: 0.3093
Epoch 20/100

Epoch 00020: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 2s 504ms/step - loss: 0.8716 - accuracy: 0.3208 - val_loss:
0.8196 - val_accuracy: 0.3093

```
0.8196 - val_accuracy: 0.3093
Epoch 21/100

Epoch 00021: LearningRateScheduler reducing learning rate to 7.812499825377017e-05.
3/3 [==============================] - 2s 504ms/step - loss: 0.8710 - accuracy: 0.3212 - val_loss:
0.8191 - val_accuracy: 0.3086
Epoch 22/100

Epoch 00022: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 2s 507ms/step - loss: 0.8705 - accuracy: 0.3216 - val_loss:
0.8189 - val_accuracy: 0.3086
Epoch 23/100

Epoch 00023: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 2s 506ms/step - loss: 0.8702 - accuracy: 0.3222 - val_loss:
0.8186 - val_accuracy: 0.3101
Epoch 24/100

Epoch 00024: LearningRateScheduler reducing learning rate to 3.9062499126885086e-05.
3/3 [==============================] - 2s 508ms/step - loss: 0.8699 - accuracy: 0.3225 - val_loss:
0.8183 - val_accuracy: 0.3108
Epoch 25/100

Epoch 00025: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 2s 510ms/step - loss: 0.8697 - accuracy: 0.3229 - val_loss:
0.8182 - val_accuracy: 0.3123
Epoch 26/100

Epoch 00026: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 2s 515ms/step - loss: 0.8695 - accuracy: 0.3231 - val_loss:
0.8181 - val_accuracy: 0.3131
Epoch 27/100

Epoch 00027: LearningRateScheduler reducing learning rate to 1.9531249563442543e-05.
3/3 [==============================] - 2s 510ms/step - loss: 0.8694 - accuracy: 0.3231 - val_loss:
0.8180 - val_accuracy: 0.3138
Epoch 28/100

Epoch 00028: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
3/3 [==============================] - 2s 518ms/step - loss: 0.8692 - accuracy: 0.3231 - val_loss:
0.8179 - val_accuracy: 0.3138
Epoch 29/100

Epoch 00029: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
3/3 [==============================] - 2s 499ms/step - loss: 0.8692 - accuracy: 0.3232 - val_loss:
0.8178 - val_accuracy: 0.3138
Epoch 30/100

Epoch 00030: LearningRateScheduler reducing learning rate to 9.765624781721272e-06.
3/3 [==============================] - 2s 505ms/step - loss: 0.8691 - accuracy: 0.3232 - val_loss:
0.8178 - val_accuracy: 0.3138
Epoch 31/100

Epoch 00031: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 2s 506ms/step - loss: 0.8690 - accuracy: 0.3232 - val_loss:
0.8177 - val_accuracy: 0.3138
Epoch 32/100

Epoch 00032: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 2s 503ms/step - loss: 0.8690 - accuracy: 0.3233 - val_loss:
0.8177 - val_accuracy: 0.3138
Epoch 33/100

Epoch 00033: LearningRateScheduler reducing learning rate to 4.882812390860636e-06.
3/3 [==============================] - 2s 510ms/step - loss: 0.8689 - accuracy: 0.3233 - val_loss:
0.8177 - val_accuracy: 0.3138
Epoch 34/100

Epoch 00034: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 2s 503ms/step - loss: 0.8689 - accuracy: 0.3234 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 35/100

Epoch 00035: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 2s 505ms/step - loss: 0.8689 - accuracy: 0.3234 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 36/100
```

```
Epoch 00036: LearningRateScheduler reducing learning rate to 2.441406195430318e-06.
3/3 [==============================] - 2s 501ms/step - loss: 0.8689 - accuracy: 0.3234 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 37/100

Epoch 00037: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 2s 501ms/step - loss: 0.8689 - accuracy: 0.3234 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 38/100

Epoch 00038: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 2s 494ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 39/100

Epoch 00039: LearningRateScheduler reducing learning rate to 1.220703097715159e-06.
3/3 [==============================] - 2s 500ms/step - loss: 0.8688 - accuracy: 0.3234 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 40/100

Epoch 00040: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 2s 501ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 41/100

Epoch 00041: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 2s 513ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 42/100

Epoch 00042: LearningRateScheduler reducing learning rate to 6.103515488575795e-07.
3/3 [==============================] - 2s 491ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 43/100

Epoch 00043: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 2s 510ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 44/100

Epoch 00044: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 2s 502ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 45/100

Epoch 00045: LearningRateScheduler reducing learning rate to 3.0517577442878974e-07.
3/3 [==============================] - 2s 494ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 46/100

Epoch 00046: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 2s 499ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 47/100

Epoch 00047: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 2s 504ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 48/100

Epoch 00048: LearningRateScheduler reducing learning rate to 1.5258788721439487e-07.
3/3 [==============================] - 2s 508ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 49/100

Epoch 00049: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
3/3 [==============================] - 2s 505ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 50/100

Epoch 00050: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
3/3 [==============================] - 2s 504ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 51/100

Epoch 00051: LearningRateScheduler reducing learning rate to 7.629394360719743e-08.
```

```
3/3 [==============================] - 2s 512ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 52/100

Epoch 00052: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 2s 511ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 53/100

Epoch 00053: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 2s 508ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 54/100

Epoch 00054: LearningRateScheduler reducing learning rate to 3.814697180359872e-08.
3/3 [==============================] - 2s 504ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 55/100

Epoch 00055: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 2s 502ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 56/100

Epoch 00056: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 2s 508ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 57/100

Epoch 00057: LearningRateScheduler reducing learning rate to 1.907348590179936e-08.
3/3 [==============================] - 2s 513ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 58/100

Epoch 00058: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 2s 499ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 59/100

Epoch 00059: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 2s 503ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 60/100

Epoch 00060: LearningRateScheduler reducing learning rate to 9.53674295089968e-09.
3/3 [==============================] - 2s 509ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 61/100

Epoch 00061: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 2s 492ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 62/100

Epoch 00062: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 2s 504ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 63/100

Epoch 00063: LearningRateScheduler reducing learning rate to 4.76837147544984e-09.
3/3 [==============================] - 2s 505ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 64/100

Epoch 00064: LearningRateScheduler reducing learning rate to 2.38418573772492e-09.
3/3 [==============================] - 2s 498ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 65/100

Epoch 00065: LearningRateScheduler reducing learning rate to 2.38418573772492e-09.
3/3 [==============================] - 2s 511ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 66/100

Epoch 00066: LearningRateScheduler reducing learning rate to 2.38418573772492e-09.
3/3 [==============================] - 2s 507ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 67/100
```

```
Epoch 67/100

Epoch 00067: LearningRateScheduler reducing learning rate to 1.19209286886246e-09.
3/3 [==============================] - 2s 507ms/step - loss: 0.8688 - accuracy: 0.3235 - val_loss:
0.8176 - val_accuracy: 0.3138
Epoch 00067: early stopping
Model: "encoder_decoder_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_12 (Encoder)         multiple                  840704
_________________________________________________________________
decoder_12 (Decoder)         multiple                  638976
_________________________________________________________________
dense_12 (Dense)             multiple                  23644
=================================================================
Total params: 1,503,324
Trainable params: 1,503,324
Non-trainable params: 0
_________________________________________________________________
```

In [143]:

```
batch_size=512
units=256
```

In [144]:

```
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 decode
  D. till we reach max_length of decoder or till the model predicted word <end>:
        predicted_out,state_h,state_c=model.layers[1](dec_input,states)
        pass the predicted_out to the dense layer
        update the states=[state_h,state_c]
        And get the index of the wordcc with maximum probability of the dense layer output, using the t
        Update the input_to_decoder with current predictions
  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 200 is the max_length of the sequence
  for i in range(200):
    infe_output, state_h = model.layers[1](cur_vec,[state_h])
    infe_output = model.layers[2](infe_output)
    cur_vec = np.reshape(np.argmax(infe_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 [147]:

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

```
The Actual Output is:
Hey u called me huh.. Wat's up leh?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
u got cash card w u now.. xin wan to photostat something..
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
I registered 4 it liao... I put class 13 first den class 14...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Wat ü doing tml? Want to go out?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Hey xin ah...R we goin 4 lesson on thurs? Oh fri rite, we r attendin e theory lesson hor...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Yeah...U hav a knack 4 sms-ing me when i'm just step 4rm knockin out.Funny thing is..I actually bother 2
reply..Which usually isn't e case.Nitey then go
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
That's great news. Sorry for the late reply. Left phone in car. Have a goodnight anyways :)
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Is it times new roman font 12 double spacing?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Haha... Sorrie abit blurr liao... Havin too much lesson... So hows life for u?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Yar lor in soc face muz be v thick one...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Nite has end for another day, morning has come in a special way. May you smile like the sunny rays and l
eaves your worries at the blue blue bay.
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Really how much...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
I'm ok wif anything..and you're e bdåy girl! You decide... Haha...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
eh she gave back the test papers ah. ?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
You study wat course?
```

```
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
juz fine. opps...Hi...how 'bout u?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
yun, if i cant make it u still going?Jos told me they going mohmd sultan, cover charge 12bucks.coz my
dinner is quite late,abt 7 plus.
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
That's how pple learn rite ?
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Leona lei... I anything lor... I'm free... But not too early...
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Actual Output is:
Not yet. where to buy
The Predicted Output is:
He ang tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou
tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou tou t
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [148]:

```python
# 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
```

In [149]:

```python
average_bleu_scores=sum(bleu_scores_lst)/len(bleu_scores_lst)
print("Average BLEU score of these 1000 test data sentences is: ",average_bleu_scores)
```

```
Average BLEU score of these 1000 test data sentences is:  0.0
```

In [150]:

```python
bleu_scores_lst
```

Out[150]:

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

In [1]:

```python
from prettytable import PrettyTable
table1=PrettyTable()
```

In [2]:

```python
table1.field_names=['S.NO','MODEL','Average_Bleu_Score']
table1.add_row([1,'Simple_ManytoMany_Characterlevel_LSTM_Model',0.32245004727098137])
table1.add_row([2,'Simple_ManytoMany_Characterlevel_Bidirectional_LSTM_Model',0.289174940583241])
table1.add_row([3,'Simple_ManytoMany_Wordlevel_LSTM_Model',0.3605904404428826])
table1.add_row([4,'Simple_ManytoMany_Wordlevel_Bidirectional_LSTM_Model',0.39176783220696243])
table1.add_row([5,'Simple_ManytoMany_Wordlevel_LSTM_Model(Fasttext_Embeddings)',0.44321749874121313])
table1.add_row([6,'Simple_ManytoMany_Wordlevel_Bidirectional_LSTM_Model(Fasttext_Embeddings)',0.48304227
table1.add_row([7,'Encoder_Decoder_Wordlevel_LSTM',0.17936797861081147])
table1.add_row([8,'Encoder_Decoder_Characterlevel_LSTM',0.1318701360495755])
table1.add_row([9,'Encoder_Decoder_Attention_Wordlevel_LSTM',0.010138124061366445])
table1.add_row([10,'Encoder_Decoder_Attention_Characterlevel_LSTM',0.015891448522335924])
table1.add_row([11,'Encoder_Decoder_Attention_Wordlevel_LSTM(Fasttext_Embeddings)',0.03251957333271974])
table1.add_row([12,'Monotonic_Attention',0.19536971776642903])
```

In [3]:

```
print(table1)
```

S.NO	MODEL	Average_Bleu_Score
1	Simple_ManytoMany_Characterlevel_LSTM_Model	0.32245004727098137
2	Simple_ManytoMany_Characterlevel_Bidirectional_LSTM_Model	0.289174940583241
3	Simple_ManytoMany_Wordlevel_LSTM_Model	0.3605904404428826
4	Simple_ManytoMany_Wordlevel_Bidirectional_LSTM_Model	0.39176783220696243
5	Simple_ManytoMany_Wordlevel_LSTM_Model(Fasttext_Embeddings)	0.44321749874121313
6	Simple_ManytoMany_Wordlevel_Bidirectional_LSTM_Model(Fasttext_Embeddings)	0.48304227467563887
7	Encoder_Decoder_Wordlevel_LSTM	0.17936797861081147
8	Encoder_Decoder_Characterlevel_LSTM	0.1318701360495755
9	Encoder_Decoder_Attention_Wordlevel_LSTM	0.010138124061366445
10	Encoder_Decoder_Attention_Characterlevel_LSTM	0.015891448522335924
11	Encoder_Decoder_Attention_Wordlevel_LSTM(Fasttext_Embeddings)	0.03251957333271974
12	Monotonic_Attention	0.19536971776642903