In [1]:

```
!pip3 install plotly

import pandas as pd
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
from keras.preprocessing.sequence import pad_sequences
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
import re
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
import tensorflow as tf
# tf.compat.v1.enable_eager_execution()
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import Multiply
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNo
rmalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
import pickle

from tqdm import tqdm
import os
import tensorflow as tf
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (4.4.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from plotly
) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.7/dist-packages
(from plotly) (1.3.3)
```

In [2]:

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
import pickle
word_final=pickle.load(open("/content/drive/MyDrive/CASE STUDY 2/word_latest.pkl", "rb"))
char_final=pickle.load(open("/content/drive/MyDrive/CASE STUDY 2/char_latest.pkl", "rb"))
```

In [ ]:

```
print('Word final dataframe:',word_final.shape)
print('Char final dataframe:',char_final.shape)
```

```
Word final dataframe: (9582, 3)
Char final dataframe: (9265, 5)
```

In [ ]:

```
word_final
```

Out[ ]:

	corrupt_word	english_in	english_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 alr...	<start> Yeap You reaching We ordered some Duri...	Yeap You reaching We ordered some Durian pastr...
2	They become more ex oredi Mine is like 25 So h...	<start> They become more expensive already Min...	They become more expensive already Mine is lik...
3	I am thai what do u do	<start> I am Thai What do you do	I am Thai What do you do <end>
4	Hi How did your week go Haven heard from you f...	<start> Hi How did your week go Have not heard...	Hi How did your week go Have not heard from yo...
...	...	...	...
9577	We rare near Coca already	<start> We are near Coca already	We are near Coca already <end>
9578	Hall eleven Got lectures end forges about comp...	<start> Hall eleven Got lectures And forget ab...	Hall eleven Got lectures And forget about comp...
9579	Hall eleven Got lectures And forget about comp...	<start> Hall eleven Got lectures And forget ab...	Hall eleven Got lectures And forget about comp...
9580	I bring for yoo i am can net promise you 100 d...	<start> I bring for you I can not promise you ...	I bring for you I can not promise you 100 to w...
9581	Im bring vor you II can not promese you 100 te...	<start> I bring for you I can not promise you ...	I bring for you I can not promise you 100 to w...

9582 rows × 3 columns

In [ ]:

```
#word_final['english_in'][1]='<start> Yeap You reaching We ordered some Durian pastry alr
eady You come quick <end>'
```

In [ ]:

```
#pickle.dump(w_final, open("/content/drive/MyDrive/CASE STUDY 2/modified.pkl","wb"))
```

In [ ]:

```
char_final['corrupt_word'] = char_final['corrupt_char'].apply(preprocess_corr)
char_final['english_in'] = char_final['english_in'].apply(preprocess_eng)
char_final['english_out'] = char_final['english_out'].apply(preprocess_eng)
```

In [ ]:

```
#pickle.dump(char_final, open("/content/drive/MyDrive/CASE STUDY 2/char_latest.pkl","wb")
)
```

A. ENCODING AT WORD LEVEL

In [ ]:

```
import pickle
word_final=pickle.load(open("/content/drive/MyDrive/CASE STUDY 2/modified.pkl", "rb"))
```

In [ ]:

```
!wget https://www.dropbox.com/s/ddkmtqz01jc024u/glove.6B.100d.txt
#downloading encoding word vector
```

```
--2021-08-14 09:19:35--  https://www.dropbox.com/s/ddkmtqz01jc024u/glove.6B.100d.txt
Resolving www.dropbox.com (www.dropbox.com)... 162.125.81.18, 2620:100:6031:18::a27d:5112
Connecting to www.dropbox.com (www.dropbox.com)|162.125.81.18|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/ddkmtqz01jc024u/glove.6B.100d.txt [following]
--2021-08-14 09:19:35--  https://www.dropbox.com/s/raw/ddkmtqz01jc024u/glove.6B.100d.txt
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucf7b637404da2a2c3e2732b01f5.dl.dropboxusercontent.com/cd/0/inline/BUNa
HvtALhBJPxtCLzZkWjI1SOQDgqKvk3lOjeBG5ogc4wbUpBsZpBSAwNXYWYOasYYtovLicfRd9kW8_o3TwgpRT2_D_
8AcPjrSxYPp8gorLRLOPFO4YOWSDR-E_wWxqXcmspLcXo_DGPdGSFBmS4_5/file# [following]
--2021-08-14 09:19:35--  https://ucf7b637404da2a2c3e2732b01f5.dl.dropboxusercontent.com/c
d/0/inline/BUNaHvtALhBJPxtCLzZkWjI1SOQDgqKvk3lOjeBG5ogc4wbUpBsZpBSAwNXYWYOasYYtovLicfRd9k
W8_o3TwgpRT2_D_8AcPjrSxYPp8gorLRLOPFO4YOWSDR-E_wWxqXcmspLcXo_DGPdGSFBmS4_5/file
Resolving ucf7b637404da2a2c3e2732b01f5.dl.dropboxusercontent.com (ucf7b637404da2a2c3e2732
b01f5.dl.dropboxusercontent.com)... 162.125.81.15, 2620:100:6031:15::a27d:510f
Connecting to ucf7b637404da2a2c3e2732b01f5.dl.dropboxusercontent.com (ucf7b637404da2a2c3e
2732b01f5.dl.dropboxusercontent.com)|162.125.81.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 347116733 (331M) [text/plain]
Saving to: 'glove.6B.100d.txt'

glove.6B.100d.txt   100%[===================>] 331.04M  15.1MB/s    in 24s

2021-08-14 09:20:00 (13.8 MB/s) - 'glove.6B.100d.txt' saved [347116733/347116733]
```

In [ ]:

```
from sklearn.model_selection import train_test_split
#splitting data with 80:20 ratios
train, validation = train_test_split(word_final, test_size=0.025,stratify=None)

print(train.shape, validation.shape)
```

```
(5850, 3) (150, 3)
```

In [ ]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
token_corr = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n',char_level=False,low
er=True,oov_token=True)
token_corr.fit_on_texts(train['corrupt_word'].values)
```

In [ ]:

```
token_eng = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n',char_level=False,lowe
r=True,oov_token=True)
token_eng.fit_on_texts(train['english_in'].values)
```

In [ ]:

```
vocab_size_eng=len(token_eng.word_index.keys())
print(vocab_size_eng)

vocab_size_corr=len(token_corr.word_index.keys())
print(vocab_size_corr)
```

```
2970
7514
```

In [ ]:

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

Out[ ]:

```
(2, 2872)
```

In [ ]:

```
embeddings_index = dict()
f = open('glove.6B.100d.txt')

for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

embedding_de = np.zeros((vocab_size_eng+1, 100))

for word, i in token_eng.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_de[i] = embedding_vector
```

In [ ]:

```
embedding_de.shape
```

Out[ ]:

```
(2971, 100)
```

In [ ]:

```
embeddings_index_1 = dict()
f = open('glove.6B.100d.txt')

for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index_1[word] = coefs
f.close()

embedding_en = np.zeros((vocab_size_corr+1, 100))

for word, i in token_corr.word_index.items():
    embedding_vector = embeddings_index_1.get(word)
    if embedding_vector is not None:
        embedding_en[i] = embedding_vector
```

In [ ]:

```
embedding_en.shape
```

Out[ ]:

```
(7515, 100)
```

In [ ]:

```
#sample_list = [train,validation,token_corr,token_eng,vocab_size_eng,vocab_size_corr, embedding_de,embedding_en]
#file_name = "/content/drive/MyDrive/CASE STUDY 2/modi_word_files.pkl"

#open_file = open(file_name, "wb")
#pickle.dump(sample_list, open_file)
#open_file.close()
```

LOAD ALL STORED FILES

In [3]:

```
#opening vocab files and embeddings
import pickle

open_file = open('/content/drive/MyDrive/CASE STUDY 2/modi_word_files.pkl', "rb")
loaded_list = pickle.load(open_file)
open_file.close()

train=loaded_list[0]
validation=loaded_list[1]
token_corr=loaded_list[2]
token_eng=loaded_list[3]
vocab_size_eng=loaded_list[4]
vocab_size_corr=loaded_list[5]
embedding_de=loaded_list[6]
embedding_en=loaded_list[7]
```

6.Improvisation of Basic Model (Using Attention Model)

In [4]:

```
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):
      super().__init__()

      self.inp_vocab_size = inp_vocab_size
      self.embedding_size = embedding_size
      self.lstm_size = lstm_size
      self.input_length = input_length

      #Initialize Embedding layer
      #Intialize Encoder LSTM layer

      #Initialize Embedding layer
      self.embedding = tf.keras.layers.Embedding(input_dim =self.inp_vocab_size , output_dim=self.embedding_size , input_length= self.input_length)
```

```
      #Intialize Encoder LSTM layer
      self.lstm = tf.keras.layers.LSTM(units=self.lstm_size , return_sequences=True, ret
urn_state=True  , name = 'Encoder_LSTM')

    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 encoder_lstm
          returns -- All encoder_outputs, last time steps hidden and cell state
      '''

      input_embedding = self.embedding(input_sequence)
      encoder_output , encoder_h , encoder_c = self.lstm(input_embedding)

      return encoder_output , encoder_h , encoder_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 zero
s is of size [32,lstm_units]
      '''

      state_h= np.zeros(shape = (batch_size , self.lstm_size))
      state_c= np.zeros(shape = (batch_size , self.lstm_size))

      return [state_h,state_c]
```

In [5]:

```
class Attention(tf.keras.layers.Layer):
  '''
    Class the calculates score based on the scoring_function using Bahdanu attention mech
anism.
  '''
  def __init__(self,scoring_function, att_units):
    super().__init__()


    # Please go through the reference notebook and research paper to complete the scoring
functions

    self.scoring_function = scoring_function
    self.att_units = att_units

    if self.scoring_function=='dot':
      # Intialize variables needed for Dot score function here
      pass
    if scoring_function == 'general':
      # Intialize variables needed for General score function here
      self.W = tf.keras.layers.Dense(units = att_units)
      pass
    elif scoring_function == 'concat':
      # Intialize variables needed for Concat score function here
      self.W1 = tf.keras.layers.Dense(self.att_units)
      self.W2 = tf.keras.layers.Dense(self.att_units)
      self.V = tf.keras.layers.Dense(1)
      pass


  def call(self,decoder_hidden_state,encoder_output):
    '''
      Attention mechanism takes two inputs current step -- decoder_hidden_state and all t
he encoder_outputs.
      * Based on the scoring function we will find the score or similarity between decode
r_hidden_state and encoder_output.
        Multiply the score function with your encoder_outputs to get the context vector.
        Function returns context vector and attention weights(softmax - scores)
    '''
```

```
    #Here from the grader function, I have taken values of
    #Attention units= 32
    #batch size= 16
    #input length= 10

    if self.scoring_function == 'dot':

        decoder_hidden_state = tf.expand_dims(decoder_hidden_state , axis = 2)
        # expand dimension will add a dimension and h,d(t-1)=(n,dimension,1)=(16,10,1)

        encoder_dot_decoder = tf.matmul(encoder_output , decoder_hidden_state)
        # Multiplying decoder hidden state with all the encoder outputs to get the alpha'
s
        # (,10,32) dot (,32,1)- Last 2 dimensions-------->(,10,1), Total dimension of en
coder_dot_decoder (16,10,1)

        attention_weight = tf.keras.activations.softmax(encoder_dot_decoder)
        #passing the alpha's to a softmax fucntion for better interpretation to get atten
tion weights (Alpha's)
        #dimension are being unchanged

        weighted_attention = attention_weight * encoder_output
        #multiplying with each encoder output for each timestamp to get context vector, t
hat is weighted attention vector
        #(,,1)*(,,32)---(last dimension)----->(,,32), now final context vector is (16,10
,32)

        context_vector = tf.reduce_sum(weighted_attention , axis = 1)
        # to find the final vector, we have summed upon the axis=1, Final context vector
shape= (16,32)
        #16--> Batch size, 32--> Attention units

        pass

    elif self.scoring_function == 'general':

        encoder_weight = self.W(encoder_output)
        #defining the W parameter ( so that final dimensions can be macthed up)--Dimensio
n (d,d')

        decoder_hidden_state = tf.expand_dims(decoder_hidden_state , axis = 2)
        # expand dimension will add a dimension and h,d(t-1)=(n,dimension,1)=(16,10,1)

        encoder_dot_decoder = tf.matmul(encoder_weight , decoder_hidden_state)
        # Multiplying decoder hidden state with all the encoder weight to get the alpha's
with corrected output dimension

        attention_weight = tf.keras.activations.softmax(encoder_dot_decoder)
        #multiplying with each encoder output for each timestamp to get context vector, t
hat is weighted attention vector

        weighted_attention = attention_weight * encoder_output
        #multiplying with each encoder output for each timestamp to get context vector, t
hat is weighted attention vector
        #(,,1)*(,,32)---(last dimension)----->(,,32), now final context vector is (16,10
,32)

        context_vector = tf.reduce_sum(weighted_attention , axis = 1)
        # to find the final vector, we have summed upon the axis=1, Final context vector
shape= (16,32)
        #16--> Batch size, 32--> Attention units

        pass

    elif self.scoring_function == 'concat':

        weight_2 = self.W2(encoder_output)
        weight_1 = self.W1(tf.expand_dims(decoder_hidden_state , axis = 1))

        #setting up vectors for multiplying vectors
```

```
        final_addition  = weight_2 + weight_1
        # [h,d(t-1)*w1+ h,e(t-1)*w2 ]

        tanh_output = tf.nn.tanh(final_addition)
        #passing it through Tanh and then used to get multiply my V

        final_ = self.V(tanh_output)

        attention_weight = tf.keras.activations.softmax(final_)
        #multiplying with each encoder output for each timestamp to get context vector, t
hat is weighted attention vector

        weighted_attention = attention_weight * encoder_output
        #multiplying with each encoder output for each timestamp to get context vector, t
hat is weighted attention vector
        #(,,1)*(,,32)---(last dimension)----->(,,32), now final context vector is (16,10
,32)

        context_vector = tf.reduce_sum(weighted_attention , axis = 1)
        # to find the final vector, we have summed upon the axis=1, Final context vector
shape= (16,32)
        #16--> Batch size, 32--> Attention units

        pass

    #Returning final context and attention weights

    return context_vector , attention_weight
```

In [7]:

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

      # Initialize decoder embedding layer, LSTM and any other objects needed

      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

      #intialised as given

      self.embedding = tf.keras.layers.Embedding(input_dim=self.tar_vocab_size , output_
dim=self.embedding_dim ,input_length=self.input_length )
      #defining embedding layer
      self.lstm = tf.keras.layers.LSTM(units = self.dec_units , return_sequences=True ,
return_state=True)
      #defining lstm layer

      self.attention = Attention(self.score_fun,self.att_units)
      #defining attention and getting weights of (t-1) timestamp

      self.dense = tf.keras.layers.Dense(units = self.tar_vocab_size)
      #defining dense layer


  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_s
ize,1,embedding_dim)
      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 out
```

```
put.
      F. Return the states from step D, output from Step E, attention weights from Step -
B
    '''

    '''
    In the grader fucntion we will use following values
    tar_vocab_size=13
    embedding_dim=12
    input_length=10
    dec_units=16
    att_units=16
    batch_size=32
    '''

    embedding_decoder = self.embedding(input_to_decoder)
    #passing the input decoder to embedding layer, embedding decoder----> (32,1,12)

    context_vector , attention_weights = self.attention(state_h,encoder_output)
    #getting context vector(batch size, attention units)----->(32,16) at timestamp (t-1)
and respective weighted attention (batch size, input lenght, 1)-----> (32,10,1)

    concatenated = tf.keras.layers.concatenate([tf.expand_dims(context_vector , 1) , emb
edding_decoder] , axis=-1)
    #concatenating them to get final input to decoder
    #(,,16)+(,,12)----->(,,28)

    decoder_output , h_state , c_state = self.lstm(concatenated , initial_state = [state
_h,state_c])
    #lstm for decoder to the current input

    decoder_output = tf.squeeze(decoder_output , axis = 1)
    # final decoder----->(batch_size,16)

    output = self.dense(decoder_output)

    return output , h_state , c_state , attention_weights,context_vector
```

In [8]:

```
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().__init__()

      self.out_vocab_size = out_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

      #intialised as per given inputs

      self.attention_w = []
      self.onestepdecoder = One_Step_Decoder(self.out_vocab_size , self.embedding_dim ,
self.input_length , self.dec_units ,self.score_fun , self.att_units )


    def call(self, input_to_decoder,encoder_output,decoder_hidden_state,decoder_cell_stat
e ):

        #Initialize an empty Tensor array, that will store the outputs at each and every
time step
        #Create a tensor array as shown in the reference notebook

        #Iterate till the length of the decoder input
            # Call onestepdecoder for each token in decoder_input
            # Store the output in tensorarray
```

```
        all_outputs = tf.TensorArray(tf.float32 , size = input_to_decoder.shape[1] , nam
e = "output_array")

        #referenced from orginal notebook

        for timestep in range(input_to_decoder.shape[1]):
          #getting results of (t-1) step where t is the current time stamp

          output , decoder_hidden_state , decoder_cell_state , attention_weights , conte
xt_vector = self.onestepdecoder(input_to_decoder[:,timestep:timestep+1],encoder_output,de
coder_hidden_state,decoder_cell_state)

          all_outputs = all_outputs.write(timestep , output)


        return tf.transpose(all_outputs.stack(),[1,0,2])
```

In [9]:

```
class Attention_Based_Encoder_Decoder(tf.keras.Model):
  def __init__(self,inp_vocab_size ,inp_embedding_size , enc_units , enc_input_length , o
ut_vocab_size , out_embedding_dim ,dec_input_length , dec_units , score_fun , att_units
, batch_size):
    #Intialize objects from encoder decoder

    super().__init__()

    self.inp_vocab_size = inp_vocab_size
    self.inp_embedding_size = inp_embedding_size
    self.enc_units = enc_units
    self.enc_input_length = enc_input_length
    self.out_vocab_size = out_vocab_size
    self.out_embedding_dim = out_embedding_dim
    self.dec_input_length = dec_input_length
    self.dec_units = dec_units
    self.score_fun = score_fun
    self.att_units = att_units
    self.batch_size = batch_size

    self.encoder = Encoder(self.inp_vocab_size ,self.inp_embedding_size,self.enc_units,s
elf.enc_input_length)
    self.decoder = Decoder(self.out_vocab_size, self.out_embedding_dim, self.dec_input_l
ength, self.dec_units ,self.score_fun ,self.att_units)


  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

    encoder_sequence , target_sequence = data[0] , data[1]

    state_h , state_c = self.encoder.initialize_states(self.batch_size)
    enc_out , enc_h , enc_c = self.encoder(encoder_sequence , [state_h , state_c])

    dec_out = self.decoder(target_sequence,enc_out,enc_h,enc_c)

    return dec_out
```

In [10]:

```
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True, reductio
n = 'none')

@tf.function
def loss_function(real, pred):
    # Custom loss function that will not consider the loss for padded zeros.
```

```
    # Refer https://www.tensorflow.org/tutorials/text/nmt_with_attention
    # optimizer = tf.keras.optimizers.Adam()
    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 [11]:

```
#tokenizing the train data as done in simple encoder decoder
train_en = tknizer_ita.texts_to_sequences(train['italian'].values) # need to pass list of values
train_dec_in= tknizer_eng.texts_to_sequences(train['english_inp'].values)
train_dec_out = tknizer_eng.texts_to_sequences(train['english_out'].values)

#padding the sequence
train_en = pad_sequences(train_en,maxlen=20, dtype='int32')
train_dec_in= pad_sequences(train_dec_in, maxlen=20,dtype='int32')
train_dec_out= pad_sequences(train_dec_out, maxlen=20,dtype='int32')

ita_word=len(tknizer_ita.index_word)
eng_word=len(tknizer_eng.index_word)
```

In [35]:

```
initial_learning_rate = 0.1

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate,decay_steps = 100000,decay_rate = 0.96,staircase = True)

model = Attention_Based_Encoder_Decoder(ita_word+1,100,128,20,eng_word+1,100,20,128,'dot',128,32)
#using the dot function

optimizer = tf.keras.optimizers.SGD(learning_rate = lr_schedule,momentum=0.9)
model.compile(optimizer=optimizer,loss=loss_function)

# Fitting the model on training data

model.fit(x = [train_en,train_dec_in],y=train_dec_out,epochs=40,validation_split=0.2)
model.summary()
```

```
Epoch 1/40
91/91 [==============================] - 43s 398ms/step - loss: 1.9749 - val_loss: 1.5470
Epoch 2/40
91/91 [==============================] - 33s 364ms/step - loss: 1.2771 - val_loss: 1.1503
Epoch 3/40
91/91 [==============================] - 34s 375ms/step - loss: 0.9098 - val_loss: 0.9290
Epoch 4/40
91/91 [==============================] - 33s 362ms/step - loss: 0.6821 - val_loss: 0.7682
Epoch 5/40
91/91 [==============================] - 34s 377ms/step - loss: 0.5314 - val_loss: 0.6470
Epoch 6/40
91/91 [==============================] - 34s 374ms/step - loss: 0.4212 - val_loss: 0.5846
Epoch 7/40
91/91 [==============================] - 33s 366ms/step - loss: 0.3558 - val_loss: 0.5146
Epoch 8/40
91/91 [==============================] - 34s 375ms/step - loss: 0.3214 - val_loss: 0.5015
Epoch 9/40
91/91 [==============================] - 33s 367ms/step - loss: 0.3039 - val_loss: 0.5032
Epoch 10/40
91/91 [==============================] - 34s 373ms/step - loss: 0.3042 - val_loss: 0.5581
Epoch 11/40
91/91 [==============================] - 34s 368ms/step - loss: 0.3308 - val_loss: 0.5718
Epoch 12/40
91/91 [==============================] - 34s 374ms/step - loss: 0.3422 - val_loss: 0.5676
Epoch 13/40
91/91 [==============================] - 33s 367ms/step - loss: 0.3301 - val_loss: 0.5086
Epoch 14/40
91/91 [==============================] - 34s 373ms/step - loss: 0.2975 - val_loss: 0.4716
Epoch 15/40
```

```
91/91 [==============================] - 34s 370ms/step - loss: 0.2472 - val_loss: 0.4399
Epoch 16/40
91/91 [==============================] - 35s 379ms/step - loss: 0.1881 - val_loss: 0.3786
Epoch 17/40
91/91 [==============================] - 34s 371ms/step - loss: 0.1307 - val_loss: 0.3068
Epoch 18/40
91/91 [==============================] - 33s 368ms/step - loss: 0.0853 - val_loss: 0.2415
Epoch 19/40
91/91 [==============================] - 33s 368ms/step - loss: 0.0552 - val_loss: 0.2053
Epoch 20/40
91/91 [==============================] - 34s 370ms/step - loss: 0.0362 - val_loss: 0.1887
Epoch 21/40
91/91 [==============================] - 34s 371ms/step - loss: 0.0267 - val_loss: 0.1755
Epoch 22/40
91/91 [==============================] - 34s 369ms/step - loss: 0.0187 - val_loss: 0.1639
Epoch 23/40
91/91 [==============================] - 33s 367ms/step - loss: 0.0146 - val_loss: 0.1598
Epoch 24/40
91/91 [==============================] - 34s 371ms/step - loss: 0.0119 - val_loss: 0.1582
Epoch 25/40
91/91 [==============================] - 34s 377ms/step - loss: 0.0100 - val_loss: 0.1569
Epoch 26/40
91/91 [==============================] - 33s 367ms/step - loss: 0.0088 - val_loss: 0.1562
Epoch 27/40
91/91 [==============================] - 34s 373ms/step - loss: 0.0077 - val_loss: 0.1502
Epoch 28/40
91/91 [==============================] - 33s 366ms/step - loss: 0.0067 - val_loss: 0.1535
Epoch 29/40
91/91 [==============================] - 34s 372ms/step - loss: 0.0060 - val_loss: 0.1555
Epoch 30/40
91/91 [==============================] - 34s 368ms/step - loss: 0.0054 - val_loss: 0.1538
Epoch 31/40
91/91 [==============================] - 34s 372ms/step - loss: 0.0049 - val_loss: 0.1569
Epoch 32/40
91/91 [==============================] - 33s 367ms/step - loss: 0.0045 - val_loss: 0.1576
Epoch 33/40
91/91 [==============================] - 34s 373ms/step - loss: 0.0040 - val_loss: 0.1589
Epoch 34/40
91/91 [==============================] - 33s 364ms/step - loss: 0.0036 - val_loss: 0.1601
Epoch 35/40
91/91 [==============================] - 34s 373ms/step - loss: 0.0034 - val_loss: 0.1584
Epoch 36/40
91/91 [==============================] - 34s 375ms/step - loss: 0.0033 - val_loss: 0.1596
Epoch 37/40
91/91 [==============================] - 33s 365ms/step - loss: 0.0027 - val_loss: 0.1590
Epoch 38/40
91/91 [==============================] - 34s 376ms/step - loss: 0.0025 - val_loss: 0.1601
Epoch 39/40
91/91 [==============================] - 33s 365ms/step - loss: 0.0023 - val_loss: 0.1592
Epoch 40/40
91/91 [==============================] - 34s 374ms/step - loss: 0.0022 - val_loss: 0.1611
Model: "attention__based__encoder__decoder_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_3 (Encoder)          multiple                  25177150
_________________________________________________________________
decoder_3 (Decoder)          multiple                  12438643
=================================================================
Total params: 37,615,793
Trainable params: 37,615,793
Non-trainable params: 0
_________________________________________________________________
```

In [22]:

```
def predict(input_sentence, model):
    '''
    Takes input sentence and model instance as inputs and predicts the output.
    The prediction is done by using following steps:
    Step A. Given input sentence, preprocess the punctuations, convert the sentence into
```

```
integers using tokenizer used earlier.
    Step B. Pass the input_sequence to encoder. we get encoder_outputs, last time step hi
dden and cell state
    Step C. Initialize index of '<' as input to decoder. and encoder final states as inpu
t_states to decoder
    Step D. Till we reach max_length of decoder or till the model predicted word '>':
            pass the inputs to timestep decoder at each timestep, update the hidden state
s and get the output token
    Step E. Return the predicted sentence.
    '''
    # Tokenizing and Padding the sentence
    inputs = [token_corr.word_index.get(i, 0) for i in input_sentence]
    inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs], maxlen = 38, paddin
g = 'post')
    inputs = tf.convert_to_tensor(inputs)

    # Initializing result string and hidden states
    result = ''
    hidden = tf.zeros([1, UNITS]), tf.zeros([1, UNITS])

    # Getting Encoder outputs
    enc_out, state_h, state_c = model.encoder(inputs, hidden)
    dec_hidden = [state_h, state_c]
    dec_input = tf.expand_dims([token_eng.word_index['<start>']], 0)

    # Running loop until max length or the prediction is '>' token
    for t in range(38):
        # Getting Decoder outputs fot timestep t
        output, state_h, state_c = model.decoder.onestepdecoder(dec_input, enc_out, stat
e_h, state_c)
        # Getting token index having highest probability
        predicted_id = tf.argmax(output[0]).numpy()
        # Getting output token
        if token_eng.index_word.get(predicted_id, '') == '<end>':
            break
        else:
            result +=token_eng.index_word.get(predicted_id, '')+' '
            dec_input = tf.expand_dims([predicted_id], 0)
    # Postprocessing the result string to remove spaces between punctuations
    return result
```

In [ ]:

```
def predictor(text):
    result = predict(text, model)
    return result

validation['predictions'] = validation['corrupt_word'].apply(predictor)

#Process inputs for Bleu score
def corrected(s):
    return [s.split()]
def predicted(s):
    return s.split()

validation['corrected'] = validation['english_in'].apply(corrected)
validation['predictions'] = validation['predictions'].apply(predicted))

score = [nltk.translate.bleu_score.sentence_bleu(validation['corrected'].iloc[i], valida
tion['predictions'].iloc[i]) for i in range(len(validation))]
print('Mean Bleu score of predictions:',np.mean(score))
```

```
Mean Bleu score of predictions: 0.4226882962168873
```

Some Random Predictions

In [ ]:

```
for i in validation.values[:10]:
  print('original:',i[0])
```

```
  print('Expected:',i[1])
  print('Predicted:',' '.join(i[3]))

  print('Blue Score:',nltk.translate.bleu_score.sentence_bleu(i[1][8:].split(' '),i[3]))
  print('-'*100)
```

```
original: Ask u something  U do ve positive plus plus feeling for me
Expected: <start> Ask u something You do have positive feeling for me
Predicted: i am out of that house already i do not care you to go to catch up think i wil
l contact you on your new line from now on right
Blue Score: 0.5081327481546147
--------------------------------------------------------------------------------------
-------------
original: Ya i noe haha got 2 interview bt watever la juz try lo heez single likewise got
chose nus sci
Expected: <start> Yes I know Haha Got 2 interviews but whatever Just try Hee I also got c
hoose NUS Science
Predicted: ok did you a good time i am going to her room now after you finish you message
us to see movie now i have to miss you that is it going to be and where is lecture
Blue Score: 0.5300714512917181
--------------------------------------------------------------------------------------
-------------
original: Yup Taken oredi Thanks
Expected: <start> Yup Taken already Thanks
Predicted: you got cash card with you now xin i am not very sure
Blue Score: 0
--------------------------------------------------------------------------------------
-------------
original: Not yet Reaching So ealy Thought you hve driving Starts a 2 right
Expected: <start> Not yet Reaching So early I thought you have driving Starts at 2 right
Predicted: so later look later you reached i have got one here is 78 128mb transcend jetf
lash do you want to meet you up
Blue Score: 0.4566337854967312
--------------------------------------------------------------------------------------
-------------
original: Hey miss where universal gas constant u
Expected: <start> Hey Miss where are you
Predicted: but i can go and wear an uniform and then you can get tickets to watch turn le
ft first i feel abused my friends already haha so that i could not dial in town okay
Blue Score: 0.41113361690051975
--------------------------------------------------------------------------------------
-------------
original: Haha K I am gonna b lat too
Expected: <start> Haha OK I am going to be late too
Predicted: well is it true for you go for a lunch well with me out with her the answers f
or the past year exam papers from
Blue Score: 0.447213595499958
--------------------------------------------------------------------------------------
-------------
original: Guess u will be usin ur new live Anyway I chose nus sci you choose nus or smu 4
your biz advertisement
Expected: <start> Guess you will be using your new line Anyway I chose NUS Science You ch
oose NUS or SMU for your business admission
Predicted: ok then i will try back to then call you in the xroom
Blue Score: 0.5266403878479265
--------------------------------------------------------------------------------------
-------------
original: HiIs forty five okif possible i would like to get the bible today ard 1700 1730
pm
Expected: <start> Hi Is 45 OK If possible I would like to get the book today at around 17
00 to 1730 pm
Predicted: ok see what new year poem you saw that is it going to be and where to meet on
wednesday sorry haha
Blue Score: 0
--------------------------------------------------------------------------------------
-------------
original: Joey YOGI CARE 2 INTRO
Expected: <start> Joey Yogi care to introduce
Predicted: i am 25 male chinese malaysian
Blue Score: 0.6389431042462724
--------------------------------------------------------------------------------------
-------------
```

```
original: I'll cat come meh Y get fire one Fri me going 2 sch lei
Expected: <start> Can I come Why got free one Friday I am going to school
Predicted: yes now this day if they do not mind going for dinner but i do not know what t
o do now so boring is it book one in time
Blue Score: 0.43092381945890607
-------------------------------------------------------------------------------------------
-------------
```

FOR GENERAL SCORING FUNCTION

In [15]:

```
initial_learning_rate = 0.1

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate,decay
_steps = 100000,decay_rate = 0.96,staircase = True)

model_2 = Attention_Based_Encoder_Decoder(ita_word+1,100,128,20,eng_word+1,100,20,128,'ge
neral',128,32)
#using the dot function

optimizer = tf.keras.optimizers.SGD(learning_rate = lr_schedule,momentum=0.9)
model_2.compile(optimizer=optimizer,loss=loss_function)
```

In [16]:

```
# Fitting the model on training data
chk_pnt = tf.keras.callbacks.ModelCheckpoint("best_model_general",save_best_only = True,
save_weights_only = False)

# Fitting the model on training data

model_2.fit(x = [train_en,train_dec_in],y=train_dec_out,epochs=40,validation_split=0.2,c
allbacks=[chk_pnt])
model_2.summary()
```

```
Epoch 1/50
91/91 [==============================] - 42s 387ms/step - loss: 2.3742 - val_loss: 2.0935

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 2/50
91/91 [==============================] - 33s 362ms/step - loss: 2.0167 - val_loss: 1.8529

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 3/50
91/91 [==============================] - 32s 353ms/step - loss: 1.7662 - val_loss: 1.6484

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets
```

```
INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 4/50
91/91 [==============================] - 33s 364ms/step - loss: 1.5498 - val_loss: 1.4856

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 5/50
91/91 [==============================] - 33s 359ms/step - loss: 1.3644 - val_loss: 1.3456

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 6/50
91/91 [==============================] - 32s 352ms/step - loss: 1.1991 - val_loss: 1.2290

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 7/50
91/91 [==============================] - 33s 360ms/step - loss: 1.0526 - val_loss: 1.1075

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 8/50
91/91 [==============================] - 32s 355ms/step - loss: 0.9170 - val_loss: 1.0041

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 9/50
91/91 [==============================] - 33s 358ms/step - loss: 0.7942 - val_loss: 0.8995

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 10/50
91/91 [==============================] - 33s 362ms/step - loss: 0.6854 - val_loss: 0.8109
```

```
WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 11/50
91/91 [==============================] - 33s 366ms/step - loss: 0.5861 - val_loss: 0.7337

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 12/50
91/91 [==============================] - 34s 368ms/step - loss: 0.5000 - val_loss: 0.6574

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 13/50
91/91 [==============================] - 33s 367ms/step - loss: 0.4253 - val_loss: 0.5891

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 14/50
91/91 [==============================] - 33s 365ms/step - loss: 0.3600 - val_loss: 0.5307

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 15/50
91/91 [==============================] - 33s 360ms/step - loss: 0.3062 - val_loss: 0.4733

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 16/50
91/91 [==============================] - 34s 369ms/step - loss: 0.2606 - val_loss: 0.4295

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 17/50
91/91 [==============================] - 33s 364ms/step - loss: 0.2192 - val_loss: 0.3754

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 18/50
91/91 [==============================] - 32s 356ms/step - loss: 0.1834 - val_loss: 0.3426

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 19/50
91/91 [==============================] - 32s 356ms/step - loss: 0.1534 - val_loss: 0.3129

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 20/50
91/91 [==============================] - 33s 362ms/step - loss: 0.1291 - val_loss: 0.2863

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 21/50
91/91 [==============================] - 32s 356ms/step - loss: 0.1098 - val_loss: 0.2590

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 22/50
91/91 [==============================] - 33s 358ms/step - loss: 0.0926 - val_loss: 0.2273

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 23/50
```

Epoch 23/50
91/91 [==============================] - 33s 366ms/step - loss: 0.0769 - val_loss: 0.2088

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 24/50
91/91 [==============================] - 33s 367ms/step - loss: 0.0638 - val_loss: 0.1855

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 25/50
91/91 [==============================] - 33s 362ms/step - loss: 0.0528 - val_loss: 0.1720

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 26/50
91/91 [==============================] - 33s 367ms/step - loss: 0.0445 - val_loss: 0.1520

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 27/50
91/91 [==============================] - 33s 364ms/step - loss: 0.0376 - val_loss: 0.1432

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 28/50
91/91 [==============================] - 33s 358ms/step - loss: 0.0313 - val_loss: 0.1291

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 29/50
91/91 [==============================] - 34s 369ms/step - loss: 0.0265 - val_loss: 0.1209

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse

```
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 30/50
91/91 [==============================] - 33s 361ms/step - loss: 0.0229 - val_loss: 0.1135

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 31/50
91/91 [==============================] - 32s 356ms/step - loss: 0.0196 - val_loss: 0.1095

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 32/50
91/91 [==============================] - 32s 356ms/step - loss: 0.0170 - val_loss: 0.1024

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 33/50
91/91 [==============================] - 33s 357ms/step - loss: 0.0149 - val_loss: 0.0969

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 34/50
91/91 [==============================] - 32s 356ms/step - loss: 0.0133 - val_loss: 0.0964

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 35/50
91/91 [==============================] - 32s 356ms/step - loss: 0.0124 - val_loss: 0.0951

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets
```

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 36/50
91/91 [==============================] - 33s 367ms/step - loss: 0.0113 - val_loss: 0.0934

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 37/50
91/91 [==============================] - 32s 355ms/step - loss: 0.0101 - val_loss: 0.0909

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 38/50
91/91 [==============================] - 33s 360ms/step - loss: 0.0091 - val_loss: 0.0886

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 39/50
91/91 [==============================] - 33s 363ms/step - loss: 0.0082 - val_loss: 0.0864

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 40/50
91/91 [==============================] - 33s 362ms/step - loss: 0.0074 - val_loss: 0.0851

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 41/50
91/91 [==============================] - 33s 364ms/step - loss: 0.0067 - val_loss: 0.0861
Epoch 42/50
91/91 [==============================] - 32s 353ms/step - loss: 0.0061 - val_loss: 0.0820

WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_conditional_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losses, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: best_model_general/assets

INFO:tensorflow:Assets written to: best_model_general/assets

Epoch 43/50
91/91 [==============================] - 32s 353ms/step - loss: 0.0056 - val_loss: 0.0841

```
Epoch 44/50
91/91 [==============================] - 33s 367ms/step - loss: 0.0053 - val_loss: 0.0819
```

```
WARNING:absl:Found untraced functions such as lstm_cell_4_layer_call_and_return_condition
al_losses, lstm_cell_4_layer_call_fn, lstm_cell_5_layer_call_and_return_conditional_losse
s, lstm_cell_5_layer_call_fn, lstm_cell_4_layer_call_fn while saving (showing 5 of 20). T
hese functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: best_model_general/assets
```

```
INFO:tensorflow:Assets written to: best_model_general/assets
```

```
Epoch 45/50
91/91 [==============================] - 33s 361ms/step - loss: 0.0054 - val_loss: 0.0908
Epoch 46/50
91/91 [==============================] - 33s 358ms/step - loss: 0.0605 - val_loss: 0.4039
Epoch 47/50
91/91 [==============================] - 33s 366ms/step - loss: 0.2964 - val_loss: 0.3833
Epoch 48/50
91/91 [==============================] - 33s 363ms/step - loss: 0.1614 - val_loss: 0.2438
Epoch 49/50
91/91 [==============================] - 33s 359ms/step - loss: 0.0767 - val_loss: 0.1750
Epoch 50/50
91/91 [==============================] - 33s 363ms/step - loss: 0.0425 - val_loss: 0.1425
```

Out[16]:

```
<tensorflow.python.keras.callbacks.History at 0x7f5b8878cf50>
```

In [ ]:

```
def predictor(text):
    result = predict(text, model)
    return result

validation['predictions_1'] = validation['corrupt_word'].apply(predictor)

#Process inputs for Bleu score
def corrected(s):
    return [s.split()]
def predicted(s):
    return s.split()

validation['corrected_1'] = validation['english_in'].apply(corrected)
validation['predictions_1'] = validation['predictions_1'].apply(predicted))

score = [nltk.translate.bleu_score.sentence_bleu(validation['corrected_1'].iloc[i], valid
ation['predictions_1'].iloc[i]) for i in range(len(validation))]
print('Mean Bleu score of predictions:',np.mean(score))
```

```
Mean Bleu score of predictions: 0.4382312902140113
```

FOR CONCAT SCORING FUNCTION

In [ ]:

```
initial_learning_rate = 0.1

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate,decay
_steps = 100000,decay_rate = 0.96,staircase = True)

model_3 = Attention_Based_Encoder_Decoder(ita_word+1,100,128,20,eng_word+1,100,20,128,'co
ncat',128,32)
#using the dot function

optimizer = tf.keras.optimizers.SGD(learning_rate = lr_schedule,momentum=0.9)
model_3.compile(optimizer=optimizer,loss=loss_function)
```

```
# Fitting the model on training data

model_3.fit(x = [train_en,train_dec_in],y=train_dec_out,epochs=40,validation_split=0.2)
model_3.summary()
```

```
Epoch 1/40
91/91 [==============================] - 55s 494ms/step - loss: 2.3665 - val_loss: 2.0644
Epoch 2/40
91/91 [==============================] - 43s 475ms/step - loss: 1.9778 - val_loss: 1.8072
Epoch 3/40
91/91 [==============================] - 43s 474ms/step - loss: 1.7139 - val_loss: 1.6081
Epoch 4/40
91/91 [==============================] - 43s 471ms/step - loss: 1.4862 - val_loss: 1.4309
Epoch 5/40
91/91 [==============================] - 43s 469ms/step - loss: 1.2894 - val_loss: 1.2806
Epoch 6/40
91/91 [==============================] - 42s 465ms/step - loss: 1.1171 - val_loss: 1.1453
Epoch 7/40
91/91 [==============================] - 43s 467ms/step - loss: 0.9626 - val_loss: 1.0296
Epoch 8/40
91/91 [==============================] - 43s 473ms/step - loss: 0.8216 - val_loss: 0.9203
Epoch 9/40
91/91 [==============================] - 44s 487ms/step - loss: 0.7036 - val_loss: 0.8242
Epoch 10/40
91/91 [==============================] - 43s 473ms/step - loss: 0.5971 - val_loss: 0.7370
Epoch 11/40
91/91 [==============================] - 43s 475ms/step - loss: 0.5035 - val_loss: 0.6511
Epoch 12/40
91/91 [==============================] - 43s 471ms/step - loss: 0.4245 - val_loss: 0.5833
Epoch 13/40
91/91 [==============================] - 44s 485ms/step - loss: 0.3598 - val_loss: 0.5161
Epoch 14/40
91/91 [==============================] - 43s 476ms/step - loss: 0.3010 - val_loss: 0.4580
Epoch 15/40
91/91 [==============================] - 43s 475ms/step - loss: 0.2521 - val_loss: 0.4068
Epoch 16/40
91/91 [==============================] - 44s 481ms/step - loss: 0.2108 - val_loss: 0.3669
Epoch 17/40
91/91 [==============================] - 43s 472ms/step - loss: 0.1750 - val_loss: 0.3251
Epoch 18/40
91/91 [==============================] - 43s 472ms/step - loss: 0.1454 - val_loss: 0.2926
Epoch 19/40
91/91 [==============================] - 43s 475ms/step - loss: 0.1196 - val_loss: 0.2564
Epoch 20/40
91/91 [==============================] - 43s 471ms/step - loss: 0.0991 - val_loss: 0.2275
Epoch 21/40
91/91 [==============================] - 42s 466ms/step - loss: 0.0817 - val_loss: 0.2072
Epoch 22/40
91/91 [==============================] - 42s 459ms/step - loss: 0.0668 - val_loss: 0.1896
Epoch 23/40
91/91 [==============================] - 42s 459ms/step - loss: 0.0546 - val_loss: 0.1734
Epoch 24/40
91/91 [==============================] - 43s 468ms/step - loss: 0.0448 - val_loss: 0.1564
Epoch 25/40
91/91 [==============================] - 43s 472ms/step - loss: 0.0378 - val_loss: 0.1422
Epoch 26/40
91/91 [==============================] - 43s 467ms/step - loss: 0.0317 - val_loss: 0.1361
Epoch 27/40
91/91 [==============================] - 43s 468ms/step - loss: 0.0268 - val_loss: 0.1226
Epoch 28/40
91/91 [==============================] - 43s 468ms/step - loss: 0.0222 - val_loss: 0.1152
Epoch 29/40
91/91 [==============================] - 43s 469ms/step - loss: 0.0193 - val_loss: 0.1072
Epoch 30/40
91/91 [==============================] - 43s 469ms/step - loss: 0.0165 - val_loss: 0.1050
Epoch 31/40
91/91 [==============================] - 43s 469ms/step - loss: 0.0146 - val_loss: 0.0999
Epoch 32/40
91/91 [==============================] - 43s 473ms/step - loss: 0.0126 - val_loss: 0.0957
Epoch 33/40
91/91 [==============================] - 43s 471ms/step - loss: 0.0109 - val_loss: 0.0919
```

```
Epoch 34/40
91/91 [==============================] - 43s 473ms/step - loss: 0.0097 - val_loss: 0.0900
Epoch 35/40
91/91 [==============================] - 43s 472ms/step - loss: 0.0087 - val_loss: 0.0865
Epoch 36/40
91/91 [==============================] - 42s 461ms/step - loss: 0.0080 - val_loss: 0.0875
Epoch 37/40
91/91 [==============================] - 42s 463ms/step - loss: 0.0074 - val_loss: 0.0791
Epoch 38/40
91/91 [==============================] - 41s 454ms/step - loss: 0.0079 - val_loss: 0.0903
Epoch 39/40
91/91 [==============================] - 43s 477ms/step - loss: 0.0077 - val_loss: 0.0871
Epoch 40/40
91/91 [==============================] - 43s 471ms/step - loss: 0.0104 - val_loss: 0.1074
Model: "attention__based__encoder__decoder_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_10 (Encoder)         multiple                  25177150
_________________________________________________________________
decoder_10 (Decoder)         multiple                  12519242
=================================================================
Total params: 37,696,392
Trainable params: 37,696,392
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
def predictor(text):
    result = predict(text, model)
    return result

validation['predictions_3'] = validation['corrupt_word'].apply(predictor)

#Process inputs for Bleu score
def corrected(s):
    return [s.split()]
def predicted(s):
    return s.split()

validation['corrected_3'] = validation['english_in'].apply(corrected)
validation['predictions_3'] = validation['predictions_3'].apply(predicted))

score = [nltk.translate.bleu_score.sentence_bleu(validation['corrected_3'].iloc[i], valid
ation['predictions_3'].iloc[i]) for i in range(len(validation))]
print('Mean Bleu score of predictions:',np.mean(score))
```

```
Mean Bleu score of predictions: 0.41657180324634446
```

7.Error Anaylsis of Best Model

PREPROCESSING BEST,WORST AND MEDIUM PREDICTIONS

CATEGORIZING DATA

In [29]:

```
best_pred=[]
medium_pred=[]
worst_pred=[]

for i in validation.values:
```

```
  pred=predict(i[1][8:],model_2)
  score=nltk.translate.bleu_score.sentence_bleu(i[1][8:],pred.split(' '))

  if score>0.45:
    best_pred.extend(pred.split(' '))

  elif score>0.30 and score<0.45:
    medium_pred.extend(pred.split(' '))
  else:
    worst_pred.extend(pred.split(' '))
```

STORING DATA

In [31]:

```
#predictions = [best_pred,medium_pred,worst_pred]
#file_name = "/content/drive/MyDrive/CASE STUDY 2/predictions.pkl"

#open_file = open(file_name, "wb")
#pickle.dump(predictions, open_file)
#open_file.close()
```

In [49]:

```
#opening vocab files and embeddings
import pickle

open_file = open('/content/drive/MyDrive/CASE STUDY 2/predictions.pkl', "rb")
loaded_list = pickle.load(open_file)
open_file.close()

best=loaded_list[0]
medium=loaded_list[1]
worst=loaded_list[2]
```

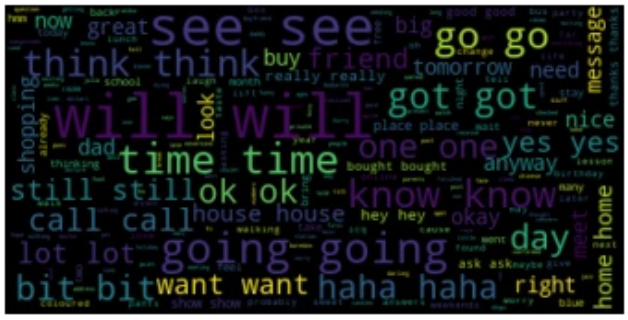
In [50]:

```
import collections

best_p=collections.Counter(best)
medium_p=collections.Counter(medium)
worst_p=collections.Counter(worst)
```

FOR BEST PREDICTIONS

In [55]:

```
import os
from os import path
from wordcloud import WordCloud
%matplotlib inline
import matplotlib.pyplot as plt

text = " ".join([(k + " ")*v for k,v in best_p.items()])

# Generate a word cloud image
wordcloud = WordCloud().generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")

# lower max_font_size
wordcloud = WordCloud(max_font_size=40).generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



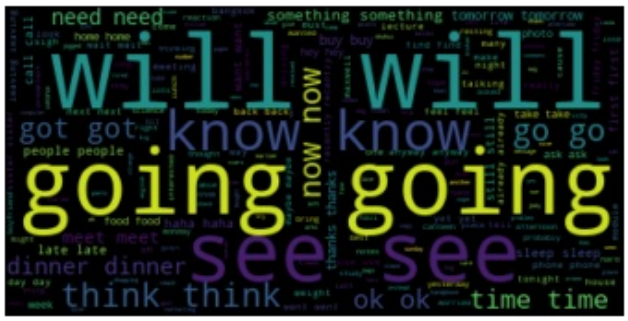
CONCLUSION:

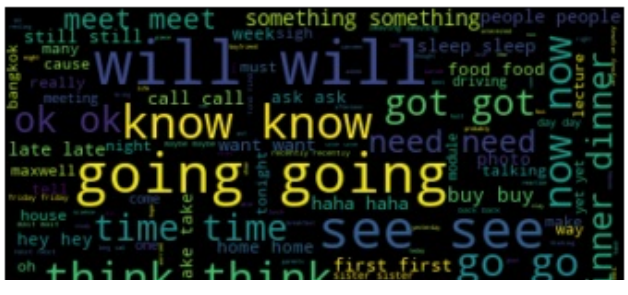
As we can see, words like going,will,see,time,know,go,one has been predicted in best case.

FOR MEDIUM GOOD PREDICTIONS

In [56]:

```
import os
from os import path
from wordcloud import WordCloud
%matplotlib inline
import matplotlib.pyplot as plt

text = " ".join([(k + " ")*v for k,v in medium_p.items()])

# Generate a word cloud image
wordcloud = WordCloud().generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")

# lower max_font_size
wordcloud = WordCloud(max_font_size=40).generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```





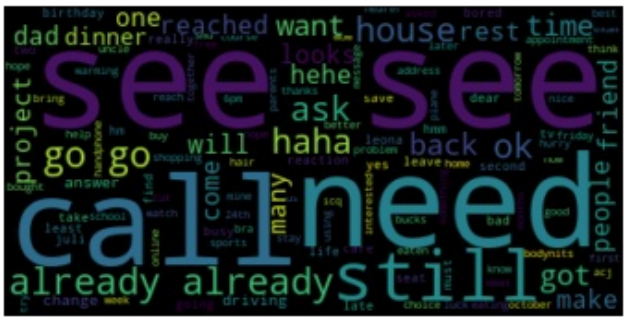
CONCLUSION:

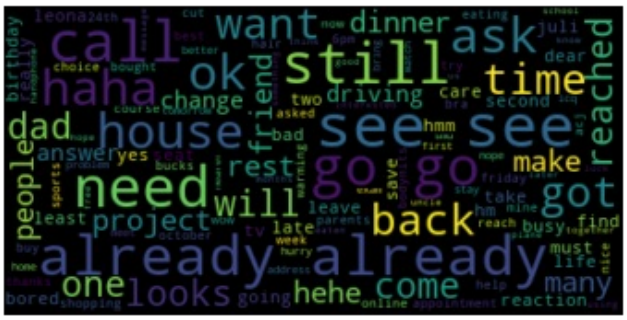
As we can see, words like going,will,know,see,think,time has been predicted in medium phase and most words are also included in the best case as well.

FOR WORST PREDICTIONS

In [57]:

```
import os
from os import path
from wordcloud import WordCloud
%matplotlib inline
import matplotlib.pyplot as plt

text = " ".join([(k + " ")*v for k,v in worst_p.items()])

# Generate a word cloud image
wordcloud = WordCloud().generate(text)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")

# lower max_font_size
wordcloud = WordCloud(max_font_size=40).generate(text)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```





CONCLUSION:

As we can see, words like call,need,already,haha,still,back has been predicted the worst, may be the words place and their spelling affected the system to recognize their true meanings.

---

8. Conclusions :

1. **For this corrupt social media text to corrected english convertions we tried 4 models respectively in different**

cases and got different results.

2. For basic encoder-decoder model, we just got 25-27% of bleu score, may be there were big sentences as well and due to augmentation, machine would have not recognized words.
3. Now we have used attention seq-2-seq models in our case study to better our results.

   3.a: During Dot scoring fucntion we got bleu score of 0.42, that is 70% higher than basic results.

   3.b: Best results is given by general scoring funciton, with 0.44 bleu score.

   3.c: And least but better results given by concat scoring function: 0.41
4. We can conclude, attention models are significant better than basic models while translation of text from one form to another and it worked for our case study too.

SUMMARY TABLE

In [60]:

```
!pip install tabletext
```

```
Collecting tabletext
  Downloading tabletext-0.1.tar.gz (6.1 kB)
Building wheels for collected packages: tabletext
  Building wheel for tabletext (setup.py) ... done
  Created wheel for tabletext: filename=tabletext-0.1-py3-none-any.whl size=6022 sha256=9
e94e351f39a917d1d8df0174e29f9040ceafeaaa5e570a0fa7e659b5629572c
  Stored in directory: /root/.cache/pip/wheels/cc/ae/ab/697f6cd9887c63663da889f796c2c7ea2
80bc407b16f6fd081
Successfully built tabletext
Installing collected packages: tabletext
Successfully installed tabletext-0.1
```

In [63]:

```
import tabletext

data = [['MODELS','BLEU SCORES'],
        ['BASIC-ENCODER-DECODER',27.2433],
        ['DOT SCORING ATTENTION MODEL',42.3978],
        ['GENERAL SCORING ATTENTION MODEL',43.9627],
        ['CONCAT SCORING ATTENTION MODEL',41.3833]
        ]

print (tabletext.to_text(data))
```

MODELS	BLEU SCORES
BASIC-ENCODER-DECODER	27.2433
DOT SCORING ATTENTION MODEL	42.3978
GENERAL SCORING ATTENTION MODEL	43.9627
CONCAT SCORING ATTENTION MODEL	41.3833