In [ ]:

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

In [ ]:

```
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, 2)
Char final dataframe: (9265, 2)
```

In [ ]:

```
word_final.head()
```

Out[ ]:

	corrupt_word	english_word
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 [ ]:

```
#adding <start> tag at english input and <end> tag at english output
word_final['english_in'] = '<start> ' + word_final['english_word'].astype(str)
word_final['english_out'] = word_final['english_word'].astype(str) + ' <end>'

word_final = word_final.drop(['english_word'], axis=1)
word_final.head() #printing final word_dataframe
```

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 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>
4	Hi! How did your week go? Haven heard from you...	<start> Hi! How did your week go? Haven't hear...	Hi! How did your week go? Haven't heard from y...

In [ ]:

```
import re
def decontractions(phrase):
    """decontracted takes text and convert contractions into natural form.
     ref: https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python/47091490#47091490"""
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"won\’t", "will not", phrase)
```

```
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)

    phrase = re.sub(r"n\’t", " not", phrase)
    phrase = re.sub(r"\’re", " are", phrase)
    phrase = re.sub(r"\’s", " is", phrase)
    phrase = re.sub(r"\’d", " would", phrase)
    phrase = re.sub(r"\’ll", " will", phrase)
    phrase = re.sub(r"\’t", " not", phrase)
    phrase = re.sub(r"\’ve", " have", phrase)
    phrase = re.sub(r"\’m", " am", phrase)

    return phrase

def preprocess_corr(text):

    # use this function to remove the contractions: https://gist.github.com/anandborad/d4
10a49a493b56dace4f814ab5325bbd
    # remove all the spacial characters: except space ' '
    text = decontractions(text)
    text = re.sub('[^A-Za-z0-9 ]+', '', text)
    return text


def preprocess_eng(text):

    # remove the words betweent brakets ()
    # remove these characters: {'$', ')', '?', '"', '’', '.',  '°', '!', ';', '/', "'",
'€', '%', ':', ',', '('}
    # replace these spl characters with space: '\u200b', '\xa0', '-', '/'
    # we have found these characters after observing the data points, feel free to explor
e more and see if you can do find more
    # you are free to do more proprocessing
    # note that the model will learn better with better preprocessed data

    text = decontractions(text)
    text = re.sub('[$)\?"’.°!;\'€%:,(/]', '', text)
    text = re.sub('\u200b', ' ', text)
    text = re.sub('\xa0', ' ', text)
    text = re.sub('-', ' ', text)
    return text
```

In [ ]:

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

In [ ]:

```
word_final.head()
```

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

	corrupt_word	english_in	english_out
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...

In [ ]:

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

In [ ]:

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

A. Encoding Word texts

In [ ]:

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

In [ ]:

```
word_final['english_in'][0]
```

Out[ ]:

```
'<start> Do you want me to reserve seat for you or not <end>'
```

In [ ]:

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

```
--2021-08-13 12:03:24--  https://www.dropbox.com/s/ddkmtqz01jc024u/glove.6B.100d.txt
Resolving www.dropbox.com (www.dropbox.com)... 162.125.2.18, 2620:100:6017:18::a27d:212
Connecting to www.dropbox.com (www.dropbox.com)|162.125.2.18|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/ddkmtqz01jc024u/glove.6B.100d.txt [following]
--2021-08-13 12:03:24--  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://ucd70f5324ecb18d416f52360562.dl.dropboxusercontent.com/cd/0/inline/BUJ3
7QaenFAcDghBrTUmsjGt4-qgodkbozpXxgigj4EkeO8B6B32iMXnvM4JHo6HU5BgaCf6EFRmMTWwGKFpek_4fhMvY
FEfh46LGuoEfkpiNbunkj0lVyhM1Puh8UqmCv-B_RC2Q2ToBLo9qFkcJkjK/file# [following]
--2021-08-13 12:03:24--  https://ucd70f5324ecb18d416f52360562.dl.dropboxusercontent.com/c
d/0/inline/BUJ37QaenFAcDghBrTUmsjGt4-qgodkbozpXxgigj4EkeO8B6B32iMXnvM4JHo6HU5BgaCf6EFRmMT
WwGKFpek_4fhMvYFEfh46LGuoEfkpiNbunkj0lVyhM1Puh8UqmCv-B_RC2Q2ToBLo9qFkcJkjK/file
Resolving ucd70f5324ecb18d416f52360562.dl.dropboxusercontent.com (ucd70f5324ecb18d416f523
60562.dl.dropboxusercontent.com)... 162.125.2.15, 2620:100:6017:15::a27d:20f
Connecting to ucd70f5324ecb18d416f52360562.dl.dropboxusercontent.com (ucd70f5324ecb18d416
f52360562.dl.dropboxusercontent.com)|162.125.2.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 347116733 (331M) [text/plain]
Saving to: ‘glove.6B.100d.txt.1’

glove.6B.100d.txt.1 100%[===================>] 331.04M  60.8MB/s    in 5.8s

2021-08-13 12:03:31 (57.3 MB/s) - ‘glove.6B.100d.txt.1’ saved [347116733/347116733]
```

In [ ]:

```
from sklearn.model_selection import train_test_split
#splitting data with 90:10 ratios
train, validation = train_test_split(word_final, test_size=0.1)
```

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

```
(8671, 3) (964, 3)
```

In [ ]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
token = Tokenizer()
token.fit_on_texts(train['corrupt_word'].values)
```

In [ ]:

```
token_eng = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n')
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.word_index.keys())
print(vocab_size_corr)
```

```
2954
10486
```

In [ ]:

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

Out[ ]:

```
(1, 2896)
```

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_matrix = 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_matrix[i] = embedding_vector
```

In [ ]:

```
embedding_matrix.shape
```

Out[ ]:

```
(2955, 100)
```

In [ ]:

```
#sample_list = [train,validation,token,token_eng,vocab_size_eng,vocab_size_corr, embeddin
g_matrix]
#file_name = "/content/drive/MyDrive/CASE STUDY 2/files_adv.pkl"

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

In [ ]:

```
#opening vocab files and embeddings
import pickle
word_final=pickle.load(open("/content/drive/MyDrive/CASE STUDY 2/word_pre_adv.pkl", "rb")
)
open_file = open('/content/drive/MyDrive/CASE STUDY 2/files_adv.pkl', "rb")

loaded_list = pickle.load(open_file)
open_file.close()

train=loaded_list[0]
validation=loaded_list[1]
token=loaded_list[2]
token_eng=loaded_list[3]
vocab_size_eng=loaded_list[4]
vocab_size_corr=loaded_list[5]
embedding_matrix=loaded_list[6]
```

B. Encoding Character texts

In [ ]:

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

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

```
(25882, 3) (2876, 3)
```

In [ ]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
char_t=Tokenizer(num_words=None,char_level=True,oov_token='UNK',lower=False)
char_t.fit_on_texts(word_final['corrupt_word'])
```

In [ ]:

```
char_e=Tokenizer(num_words=None,char_level=True,oov_token='UNK',lower=False)
char_e.fit_on_texts(word_final['english_in'])
```

In [ ]:

```
seq=char_t.texts_to_sequences(word_final['corrupt_word'])
print(word_final['corrupt_word'][0])
len(seq[0])
```

```
U wan me to chop seat 4 u nt
```

Out[ ]:

```
28
```

In [ ]:

```
seq_e=char_e.texts_to_sequences(word_final['english_in'])
print(word_final['english_in'][0])
len(seq_e[0])
```

```
<start> Do you want me to reserve seat for you or not
```

Out[ ]:

```
53
```

In [ ]:

```
char_pd=pad_sequences(seq,maxlen=159,padding='post')
print(seq[0][:159])
char_pd[0][:159]
```

```
[58, 2, 18, 6, 7, 2, 16, 3, 2, 5, 4, 2, 19, 10, 4, 22, 2, 11, 3, 6, 5, 2, 56, 2, 12, 2, 7,
5]
```

Out[ ]:

```
array([58,  2, 18,  6,  7,  2, 16,  3,  2,  5,  4,  2, 19, 10,  4, 22,  2,
       11,  3,  6,  5,  2, 56,  2, 12,  2,  7,  5,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0], dtype=int32)
```

In [ ]:

```
char_pd_e=pad_sequences(seq_e,maxlen=190,padding='post')
print(seq[0][:190])
char_pd[0][:190]
```

```
[58, 2, 18, 6, 7, 2, 16, 3, 2, 5, 4, 2, 19, 10, 4, 22, 2, 11, 3, 6, 5, 2, 56, 2, 12, 2, 7,
5]
```

Out[ ]:

```
array([58,  2, 18,  6,  7,  2, 16,  3,  2,  5,  4,  2, 19, 10,  4, 22,  2,
       11,  3,  6,  5,  2, 56,  2, 12,  2,  7,  5,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0], dtype=int32)
```

In [ ]:

```
final_data=np.array(char_pd)
final_data.shape
```

Out[ ]:

```
(28758, 159)
```

In [ ]:

```
final_data_e=np.array(char_pd_e)
final_data_e.shape
```

Out[ ]:

```
(28758, 190)
```

In [ ]:

```
vocab_char_eng=len(char_e.word_index.keys())
print(vocab_char_eng)

vocab_char_corr=len(char_t.word_index.keys())
print(vocab_char_corr)
```

```
79
64
```

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_matrix_1 = np.zeros((vocab_char_eng+1, 100))

for word, i in char_e.word_index.items():
    embedding_vector = embeddings_index_1.get(word)
    if embedding_vector is not None:
        embedding_matrix_1[i] = embedding_vector
```

In [ ]:

```
embedding_matrix_1.shape
```

Out[ ]:

```
(80, 100)
```

5.Training Basic Model And Debugging (Simple Encoder And Decoder)

A. Word Level

In [ ]:

```
class Encoder(tf.keras.layers.Layer):
    def __init__(self, vocab_size, embedding_dim, input_length, enc_units):
        super().__init__()
        self.vocab_size = vocab_size
        self.embedding_dim = embedding_dim
        self.input_length = input_length
        self.enc_units= enc_units
        self.lstm_output = 0
        self.lstm_state_h=0
        self.lstm_state_c=0

    def build(self, input_shape):
        self.embedding = Embedding(input_dim=self.vocab_size, output_dim=self.embedding_
dim, input_length=self.input_length,
                                   mask_zero=True, name="embedding_layer_encoder")
        self.lstm = LSTM(self.enc_units, return_state=True, return_sequences=True, name=
"Encoder_LSTM")

    def call(self, input_sentances,states):

        input_embedd                                = self.embedding(input_sentances)
        self.lstm_output, self.lstm_state_h,self.lstm_state_c = self.lstm(input_embedd,i
nitial_state=states)
        return self.lstm_output, self.lstm_state_h,self.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 zero
s is of size [32,lstm_units]
      '''
```

```
        return tf.zeros([batch_size, self.enc_units]), tf.zeros([batch_size, self.enc_units])


class Decoder(tf.keras.layers.Layer):
    def __init__(self, vocab_size, embedding_dim, input_length, dec_units):
        super().__init__()
        self.vocab_size = vocab_size
        self.embedding_dim = 100
        self.dec_units = dec_units
        self.input_length = input_length
        # we are using embedding_matrix and not training the embedding layer
        self.embedding = Embedding(input_dim=self.vocab_size, output_dim=self.embedding_dim, input_length=self.input_length,
                                   mask_zero=True, name="embedding_layer_decoder", weights=[embedding_matrix], trainable=False)
        self.lstm = LSTM(self.dec_units, return_sequences=True, return_state=True, name="Encoder_LSTM")

    def call(self, target_sentances, state_h, state_c):
        target_embedd           = self.embedding(target_sentances)
        lstm_output, decoder_final_state_h,decoder_final_state_c  = self.lstm(target_embedd, initial_state=[state_h, state_c])
        return lstm_output,decoder_final_state_h,decoder_final_state_c
```

In [ ]:

```
class Dataset:
    def __init__(self, data, tknizer_ita, tknizer_eng, max_len):
        self.encoder_inps = data['corrupt_word'].values
        self.decoder_inps = data['english_in'].values
        self.decoder_outs = data['english_out'].values
        self.tknizer_eng = tknizer_eng
        self.tknizer_ita = tknizer_ita
        self.max_len = max_len

    def __getitem__(self, i):
        self.encoder_seq = self.tknizer_ita.texts_to_sequences([self.encoder_inps[i]]) # need to pass list of values
        self.decoder_inp_seq = self.tknizer_eng.texts_to_sequences([self.decoder_inps[i]])
        self.decoder_out_seq = self.tknizer_eng.texts_to_sequences([self.decoder_outs[i]])

        self.encoder_seq = pad_sequences(self.encoder_seq, maxlen=self.max_len, dtype='int32', padding='post')
        self.decoder_inp_seq = pad_sequences(self.decoder_inp_seq, maxlen=self.max_len, dtype='int32', padding='post')
        self.decoder_out_seq = pad_sequences(self.decoder_out_seq, maxlen=self.max_len, dtype='int32', padding='post')
        return self.encoder_seq, self.decoder_inp_seq, self.decoder_out_seq

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


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


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

        batch = [np.squeeze(np.stack(samples, axis=1), axis=0) for samples in zip(*data)
```

```
]
        # we are creating data like ([italian, english_inp], english_out) these are alrea
dy converted into seq
        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 [ ]:

```
train_dataset = Dataset(train, token, token_eng, 38)
test_dataset  = Dataset(validation, token, token_eng, 38)

train_dataloader = Dataloder(train_dataset, batch_size=64)
test_dataloader = Dataloder(test_dataset, batch_size=64)


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

```
(64, 38) (64, 38) (64, 38)
```

In [ ]:

```
# this is the same model we have given in the other reference notebook
class MyModel(Model):
    def __init__(self, encoder_inputs_length,decoder_inputs_length, output_vocab_size):
        super().__init__() # https://stackoverflow.com/a/27134600/4084039
        self.encoder = Encoder(vocab_size=vocab_size_corr+1, embedding_dim=50, input_len
gth=encoder_inputs_length, enc_units=64)
        self.decoder = Decoder(vocab_size=vocab_size_eng+1, embedding_dim=100, input_len
gth=decoder_inputs_length, dec_units=64)
        self.dense   = Dense(output_vocab_size+1, activation='softmax')
        self.batch_size= 64


    def call(self, data):
        input,output = data[0], data[1]
        encoder_output, encoder_h, encoder_c = self.encoder(input, self.encoder.initiali
ze_states(self.batch_size))
        decoder_output,a,b                       = self.decoder(output, encoder_h, encod
er_c)
        output                                = self.dense(decoder_output)
        return output
```

In [ ]:

```
class LossHistory(tf.keras.callbacks.Callback):

    def on_train_begin(self, logs={}):
        ## on begin of training, we are creating a instance varible called history
        ## it is a dict with keys [loss, acc, val_loss, val_acc]
        self.history={'loss': [],'val_loss': []}

    def on_epoch_end(self, epoch, logs={}):
        ## on end of each epoch, we will get logs and update the self.history dict
        self.history['loss'].append(logs.get('loss'))
        if logs.get('val_loss', -1) != -1:
            self.history['val_loss'].append(logs.get('val_loss'))

history_own=LossHistory()
chk=tf.keras.callbacks.ModelCheckpoint('basic_model', monitor='val_loss', verbose=0, sav
e_best_only=True,\
                                       save_weights_only=False, mode='auto', save_freq='epoc
h',\
                                       options=None)
```

In [ ]:

```
model  = MyModel(encoder_inputs_length=35,decoder_inputs_length=38,output_vocab_size=voc
ab_size_eng)
optimizer = tf.keras.optimizers.Adam()
model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy')
train_steps=train.shape[0]//128
valid_steps=validation.shape[0]//128
```

In [ ]:

```
#model.load_weights('/content/drive/MyDrive/CASE STUDY 2/basic_model')
```

Out[ ]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fdfd9c20510>
```

In [ ]:

```
#running model for 50 epochs

model.fit_generator(train_dataloader, steps_per_epoch=train_steps, epochs=100, validation
_data=train_dataloader,\
                    validation_steps=valid_steps,callbacks=[history_own])
```

```
Epoch 1/100
67/67 [==============================] - 2s 26ms/step - loss: 2.2206 - val_loss: 2.1987
Epoch 2/100
67/67 [==============================] - 2s 26ms/step - loss: 2.1764 - val_loss: 2.1768
Epoch 3/100
67/67 [==============================] - 2s 26ms/step - loss: 2.1775 - val_loss: 2.1533
Epoch 4/100
67/67 [==============================] - 2s 27ms/step - loss: 2.1308 - val_loss: 2.1215
Epoch 5/100
67/67 [==============================] - 2s 26ms/step - loss: 2.0908 - val_loss: 2.0947
Epoch 6/100
67/67 [==============================] - 2s 26ms/step - loss: 2.0828 - val_loss: 2.0676
Epoch 7/100
67/67 [==============================] - 2s 27ms/step - loss: 2.0547 - val_loss: 2.0446
Epoch 8/100
67/67 [==============================] - 2s 26ms/step - loss: 2.0135 - val_loss: 2.0202
Epoch 9/100
67/67 [==============================] - 2s 28ms/step - loss: 1.9812 - val_loss: 1.9991
Epoch 10/100
67/67 [==============================] - 2s 28ms/step - loss: 1.9829 - val_loss: 1.9788
Epoch 11/100
67/67 [==============================] - 2s 28ms/step - loss: 1.9745 - val_loss: 1.9548
Epoch 12/100
67/67 [==============================] - 2s 29ms/step - loss: 1.9641 - val_loss: 1.9351
Epoch 13/100
67/67 [==============================] - 2s 29ms/step - loss: 1.9469 - val_loss: 1.9191
Epoch 14/100
67/67 [==============================] - 2s 29ms/step - loss: 1.9111 - val_loss: 1.9041
Epoch 15/100
67/67 [==============================] - 2s 29ms/step - loss: 1.9048 - val_loss: 1.8873
Epoch 16/100
67/67 [==============================] - 2s 29ms/step - loss: 1.8864 - val_loss: 1.8739
Epoch 17/100
67/67 [==============================] - 2s 28ms/step - loss: 1.8676 - val_loss: 1.8559
Epoch 18/100
67/67 [==============================] - 2s 29ms/step - loss: 1.8690 - val_loss: 1.8384
Epoch 19/100
67/67 [==============================] - 2s 29ms/step - loss: 1.8312 - val_loss: 1.8241
Epoch 20/100
67/67 [==============================] - 2s 28ms/step - loss: 1.8053 - val_loss: 1.8034
Epoch 21/100
67/67 [==============================] - 2s 27ms/step - loss: 1.8058 - val_loss: 1.7837
Epoch 22/100
67/67 [==============================] - 2s 27ms/step - loss: 1.7869 - val_loss: 1.7690
Epoch 23/100
67/67 [==============================] - 2s 27ms/step - loss: 1.7583 - val_loss: 1.7513
Epoch 24/100
67/67 [==============================] - 2s 27ms/step - loss: 1.7445 - val_loss: 1.7285
Epoch 25/100
```

```
67/67 [==============================] - 2s 26ms/step - loss: 1.7148 - val_loss: 1.7152
Epoch 26/100
67/67 [==============================] - 2s 27ms/step - loss: 1.7022 - val_loss: 1.7024
Epoch 27/100
67/67 [==============================] - 2s 27ms/step - loss: 1.6930 - val_loss: 1.6872
Epoch 28/100
67/67 [==============================] - 2s 27ms/step - loss: 1.6646 - val_loss: 1.6699
Epoch 29/100
67/67 [==============================] - 2s 26ms/step - loss: 1.6631 - val_loss: 1.6579
Epoch 30/100
67/67 [==============================] - 2s 26ms/step - loss: 1.6330 - val_loss: 1.6454
Epoch 31/100
67/67 [==============================] - 2s 26ms/step - loss: 1.6320 - val_loss: 1.6276
Epoch 32/100
67/67 [==============================] - 2s 27ms/step - loss: 1.6106 - val_loss: 1.6121
Epoch 33/100
67/67 [==============================] - 2s 27ms/step - loss: 1.5880 - val_loss: 1.5937
Epoch 34/100
67/67 [==============================] - 2s 27ms/step - loss: 1.5846 - val_loss: 1.5732
Epoch 35/100
67/67 [==============================] - 2s 27ms/step - loss: 1.5458 - val_loss: 1.5583
Epoch 36/100
67/67 [==============================] - 2s 26ms/step - loss: 1.5461 - val_loss: 1.5450
Epoch 37/100
67/67 [==============================] - 2s 25ms/step - loss: 1.5431 - val_loss: 1.5360
Epoch 38/100
67/67 [==============================] - 2s 27ms/step - loss: 1.5009 - val_loss: 1.5240
Epoch 39/100
67/67 [==============================] - 2s 29ms/step - loss: 1.5171 - val_loss: 1.5053
Epoch 40/100
67/67 [==============================] - 2s 28ms/step - loss: 1.5052 - val_loss: 1.4944
Epoch 41/100
67/67 [==============================] - 2s 29ms/step - loss: 1.5022 - val_loss: 1.4803
Epoch 42/100
67/67 [==============================] - 2s 28ms/step - loss: 1.4740 - val_loss: 1.4702
Epoch 43/100
67/67 [==============================] - 2s 28ms/step - loss: 1.4576 - val_loss: 1.4577
Epoch 44/100
67/67 [==============================] - 2s 28ms/step - loss: 1.4494 - val_loss: 1.4415
Epoch 45/100
67/67 [==============================] - 2s 29ms/step - loss: 1.4438 - val_loss: 1.4290
Epoch 46/100
67/67 [==============================] - 2s 29ms/step - loss: 1.4303 - val_loss: 1.4173
Epoch 47/100
67/67 [==============================] - 2s 28ms/step - loss: 1.3999 - val_loss: 1.4047
Epoch 48/100
67/67 [==============================] - 2s 28ms/step - loss: 1.3929 - val_loss: 1.3932
Epoch 49/100
67/67 [==============================] - 2s 29ms/step - loss: 1.3966 - val_loss: 1.3831
Epoch 50/100
67/67 [==============================] - 2s 27ms/step - loss: 1.3763 - val_loss: 1.3679
Epoch 51/100
67/67 [==============================] - 2s 26ms/step - loss: 1.3371 - val_loss: 1.3541
Epoch 52/100
67/67 [==============================] - 2s 27ms/step - loss: 1.3760 - val_loss: 1.3509
Epoch 53/100
67/67 [==============================] - 2s 27ms/step - loss: 1.3336 - val_loss: 1.3368
Epoch 54/100
67/67 [==============================] - 2s 27ms/step - loss: 1.3361 - val_loss: 1.3280
Epoch 55/100
67/67 [==============================] - 2s 27ms/step - loss: 1.3102 - val_loss: 1.3135
Epoch 56/100
67/67 [==============================] - 2s 26ms/step - loss: 1.3040 - val_loss: 1.3069
Epoch 57/100
67/67 [==============================] - 2s 26ms/step - loss: 1.2967 - val_loss: 1.2923
Epoch 58/100
67/67 [==============================] - 2s 27ms/step - loss: 1.2849 - val_loss: 1.2791
Epoch 59/100
67/67 [==============================] - 2s 27ms/step - loss: 1.2697 - val_loss: 1.2684
Epoch 60/100
67/67 [==============================] - 2s 27ms/step - loss: 1.2757 - val_loss: 1.2624
Epoch 61/100
```

```
67/67 [==============================] - 2s 26ms/step - loss: 1.2595 - val_loss: 1.2482
Epoch 62/100
67/67 [==============================] - 2s 27ms/step - loss: 1.2487 - val_loss: 1.2411
Epoch 63/100
67/67 [==============================] - 2s 27ms/step - loss: 1.2558 - val_loss: 1.2360
Epoch 64/100
67/67 [==============================] - 2s 26ms/step - loss: 1.2400 - val_loss: 1.2303
Epoch 65/100
67/67 [==============================] - 2s 26ms/step - loss: 1.2117 - val_loss: 1.2191
Epoch 66/100
67/67 [==============================] - 2s 26ms/step - loss: 1.2112 - val_loss: 1.2128
Epoch 67/100
67/67 [==============================] - 2s 26ms/step - loss: 1.1994 - val_loss: 1.2046
Epoch 68/100
67/67 [==============================] - 2s 27ms/step - loss: 1.1942 - val_loss: 1.1919
Epoch 69/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1979 - val_loss: 1.1892
Epoch 70/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1696 - val_loss: 1.1895
Epoch 71/100
67/67 [==============================] - 2s 29ms/step - loss: 1.1728 - val_loss: 1.1692
Epoch 72/100
67/67 [==============================] - 2s 29ms/step - loss: 1.1591 - val_loss: 1.1638
Epoch 73/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1396 - val_loss: 1.1567
Epoch 74/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1464 - val_loss: 1.1489
Epoch 75/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1461 - val_loss: 1.1432
Epoch 76/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1256 - val_loss: 1.1332
Epoch 77/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1224 - val_loss: 1.1243
Epoch 78/100
67/67 [==============================] - 2s 28ms/step - loss: 1.1121 - val_loss: 1.1111
Epoch 79/100
67/67 [==============================] - 2s 29ms/step - loss: 1.0984 - val_loss: 1.1045
Epoch 80/100
67/67 [==============================] - 2s 27ms/step - loss: 1.1081 - val_loss: 1.1013
Epoch 81/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0898 - val_loss: 1.0944
Epoch 82/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0741 - val_loss: 1.0886
Epoch 83/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0773 - val_loss: 1.0779
Epoch 84/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0605 - val_loss: 1.0739
Epoch 85/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0702 - val_loss: 1.0616
Epoch 86/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0693 - val_loss: 1.0585
Epoch 87/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0347 - val_loss: 1.0544
Epoch 88/100
67/67 [==============================] - 2s 26ms/step - loss: 1.0350 - val_loss: 1.0532
Epoch 89/100
67/67 [==============================] - 2s 26ms/step - loss: 1.0301 - val_loss: 1.0482
Epoch 90/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0343 - val_loss: 1.0381
Epoch 91/100
67/67 [==============================] - 2s 26ms/step - loss: 1.0262 - val_loss: 1.0278
Epoch 92/100
67/67 [==============================] - 2s 26ms/step - loss: 1.0351 - val_loss: 1.0198
Epoch 93/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0179 - val_loss: 1.0125
Epoch 94/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0025 - val_loss: 1.0082
Epoch 95/100
67/67 [==============================] - 2s 27ms/step - loss: 1.0048 - val_loss: 1.0047
Epoch 96/100
67/67 [==============================] - 2s 28ms/step - loss: 1.0185 - val_loss: 1.0004
Epoch 97/100
```

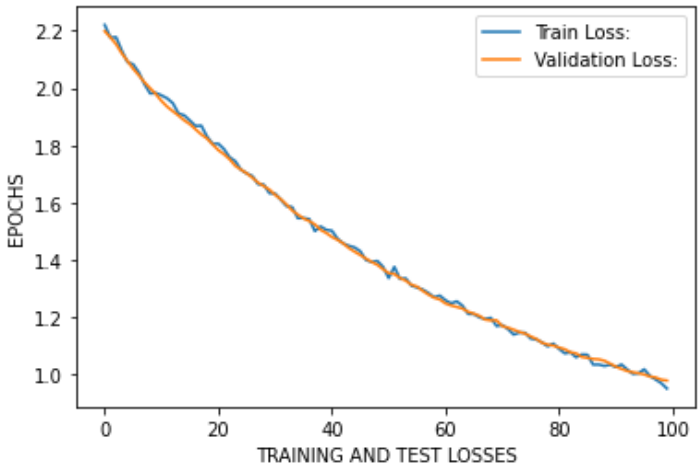
```
67/67 [==============================] - 2s 28ms/step - loss: 0.9952 - val_loss: 0.9930
Epoch 98/100
67/67 [==============================] - 2s 30ms/step - loss: 0.9829 - val_loss: 0.9900
Epoch 99/100
67/67 [==============================] - 2s 30ms/step - loss: 0.9702 - val_loss: 0.9820
Epoch 100/100
67/67 [==============================] - 2s 30ms/step - loss: 0.9513 - val_loss: 0.9794
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7fc5cab17d90>
```

In [ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

# create data
train_loss=history_own.history['loss']
val_loss=history_own.history['val_loss']

epoch=[]
for i in range(100):
    epoch.append(i)


plt.plot(epoch, train_loss, label = "Train Loss:")
plt.plot(epoch, val_loss, label = "Validation Loss:")

plt.xlabel("TRAINING AND TEST LOSSES")
plt.ylabel("EPOCHS")

plt.legend()
plt.show()
```



Predictions :

In [ ]:

```
print(tf.__version__)
```

```
2.5.0
```

In [ ]:

```
def predict(input_sentence):

  ''' This function predicts the text to a given corrupted input text'''

  #This code is used thorugh same steps used while training, preprocessing our Data

  intial_states = model.layers[0].initialize_states(1)
```

```
  inp_seq = token.texts_to_sequences([input_sentence])
  inp_seq = pad_sequences(inp_seq,padding='post',maxlen=38)

  en_outputs,state_h , state_c = model.encoder(tf.constant(inp_seq),intial_states)
  cur_vec = tf.constant([[token_eng.word_index['<start>']]])
  score = []

  #Here 38 is the max_length of the sequence

  for i in range(38):
   infe_output,state_h, state_c = model.decoder(cur_vec,state_h,state_c)
   infe_output = model.layers[2](infe_output)

   cur_vec = np.reshape(np.argmax(infe_output), (1, 1))
   score.append(token_eng.index_word[cur_vec[0][0]])

   if(score[-1]=='<end>'): #if <end> token is reached
     break

  return ' '.join(score)
```

Bleu Scoring :

In [ ]:

```
import nltk.translate.bleu_score as bleu

corrupt = validation.corrupt_word.values[:1000]
corrected = validation.english_out.values[:1000]

bleu_s = 0

for index , i in enumerate(corrupt):
  text=predict(i)
  #predicting corrected english step for each sentence
  translation = text.split()
  reference = corrected[index].split()
  bleu_s += bleu.sentence_bleu(reference, translation)
```

In [ ]:

```
print('The bleu score is:',bleu_s/1000)
```

```
The bleu score is: 0.26273701253928017
```

In [ ]:

```
corr=validation.corrupt_word.values[:10]
eng=validation.english_out.values

for a,b in enumerate(corr):
  print('The Text:',eng[a])
  print('Predicted:',predict(b))
  print('-'*100)
```

```
The Text: Huh Oh That is the wooden one right The aluminium one is cheaper <end>
Predicted: huh so is it is not going to be late i am going to school <end>
----------------------------------------------------------------------------------------------------
The Text: What time will you end then We are at OG <end>
Predicted: what time do not i got a good night <end>
----------------------------------------------------------------------------------------------------
The Text: Are you girls going to have lunch before going <end>
Predicted: you are going to be late for you <end>
----------------------------------------------------------------------------------------------------
The Text: Hi want to chat Please SMS me at 99853267 Thanks and see you <end>
Predicted: hello want to chat with you and your dog me to see you <end>
----------------------------------------------------------------------------------------------------
```

```
-------------
The Text: What time Now going to rain <end>
Predicted: what time will you be going <end>
-------------------------------------------------------------------------------------------
-------------
The Text: No no sports car for me I have gotten into lots of accidents for my love of spe
ed Furthermore it is too expensive for me already So I changed to a normal car <end>
Predicted: no my sister is not going to meet you at the way i am going to be late i am go
ing to be late i am having a bit lousy so much time do you want <end>
-------------------------------------------------------------------------------------------
-------------
The Text: I thought they said I should receive it by the 24th <end>
Predicted: eh i think i will go to go for your party <end>
-------------------------------------------------------------------------------------------
-------------
The Text: Maintenance technician How about you <end>
Predicted: how are you doing <end>
-------------------------------------------------------------------------------------------
-------------
The Text: Yes of course go back with you <end>
Predicted: yes good of the way are you <end>
-------------------------------------------------------------------------------------------
-------------
The Text: Okay then When are you coming back Have a safe trip And buy some nice clothes o
r stuff back for us Haha <end>
Predicted: okay then see you at home to take care for a few days ago i have not seen you
to take care <end>
-------------------------------------------------------------------------------------------
-------------
```