```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
import os
os.chdir('/content/drive/MyDrive')
```

Importing Libiraries

```
import numpy as np
import pandas as pd
import  string
from string import digits
import re
from sklearn.utils import  shuffle
from sklearn.model_selection import train_test_split
from keras.layers import Input,LSTM,Embedding,Dense
from keras.models import Model

import seaborn as sns
import matplotlib.pyplot as plt
```


Load The Data

```
df = pd.read_csv("Hindi_English_Truncated_Corpus.csv")
df.head(2)
```

	source	english_sentence	hindi_sentence
0	ted	politicians do not have permission to do what ...	राजनीतिज्ञों के पास जो कार्य करना चाहिए, वह कर...
1	ted	I'd like to tell you about one such child	मई आपको ऐसे ही एक बच्चे के बारे में बताना

```
df.columns
```

```
Index(['source', 'english_sentence', 'hindi_sentence'], dtype='object')
```

```
df["source"].value_counts()
```

```
tides        50000
ted          39881
```

```
indic2012    37726
Name: source, dtype: int64
```

Let's take sample data 5000

```
df =df.sample(40000)
```

```
df.shape
```

```
(40000, 3)
```

```
df.drop("source",axis=1,inplace=True)
```

```
df.shape
```

```
(40000, 2)
```

```
df[df.duplicated()]
```

	english_sentence	hindi_sentence
51416	(Laughter)	(हंसी)
24041	(Laughter)	(हँसी)
70275	(Laughter)	(हँसी)
71156	context	संदर्भ
...	...	...
6192	(Laughter)	(हँसी)
6729	See this also	यह भी देखिए
55540	(Applause)	(तालियाँ)
49232	Beginning	आरम्भ
16980	(Music)	(संगीत)

552 rows × 2 columns


```
len(df[df.duplicated()])
```

```
552
```

```
df.drop_duplicates(inplace=True)
```

```
df.shape
```

```
(39448, 2)
```

```
5000-4965
```

```
35
```

```
df.isnull().sum()
```

```
english_sentence    0
hindi_sentence      0
dtype: int64
```

Let Start Cleaning Process

```
df.reset_index(inplace=True)
```

```
df.drop("index",axis=1,inplace=True)
```

```
df.head(5)
```


	english_sentence	hindi_sentence
0	Another Rs 1.46 crore arrived in four instalme...	पाए आर माच 1995 आर नवबर 1999 के बीच चार किश...
1	You've got to just respond to the situation.	आप केवल परिस्थितियों के अनुसार कार्य कर सकते हैं
2	that the opposite may also be true.	बिल्कुल विपरीत सोच भी उतनी ही वैध हो सकती है.
3	martial art movements.”	भी कर सकता हूँ।
4	South Africa: 222	दक्षिण अफ्रीका : २२२

Data Preprocessing

English_Text_Preprocessing

```
# Here we are replacing the words
mispell_dict = {"aren't" : "are not", "can't" : "cannot", "couldn't" : "could not",
                "couldnt" : "could not", "didn't" : "did not", "doesn't" : "does not",
                "doesnt" : "does not", "don't" : "do not", "hadn't" : "had not",
                "hasn't" : "has not", "haven't" : "have not", "havent" : "have not",
```

```
                "he'd" : "he would", "he'll" : "he will", "he's" : "he is", "i'd" : "I would"
                "i'd" : "I had", "i'll" : "I will", "i'm" : "I am", "isn't" : "is not",
                "it's" : "it is", "it'll":"it will", "i've" : "I have", "let's" : "let us",
                "mightn't" : "might not", "mustn't" : "must not", "shan't" : "shall not",
                "she'd" : "she would", "she'll" : "she will", "she's" : "she is",
                "shouldn't" : "should not", "shouldnt" : "should not", "that's" : "that is",
                "thats" : "that is", "there's" : "there is", "theres" : "there is",
                "they'd" : "they would", "they'll" : "they will", "they're" : "they are",
                "theyre": "they are", "they've" : "they have", "we'd" : "we would",
                "we're" : "we are", "weren't" : "were not", "we've" : "we have",
                "what'll" : "what will", "what're" : "what are", "what's" : "what is",
                "what've" : "what have", "where's" : "where is", "who'd" : "who would",
                "who'll" : "who will", "who're" : "who are", "who's" : "who is",
                "who've" : "who have", "won't" : "will not", "wouldn't" : "would not",
                "you'd" : "you would", "you'll" : "you will", "you're" : "you are",
                "you've" : "you have", "'re": " are", "wasn't": "was not", "we'll":" will",
                "didn't": "did not", "tryin'":"trying"
                }

def deconcate(sent):
    keys = mispell_dict.keys()
    cleaned_sentence = []
    for word in sent.split(' '):
        if word in keys:
            cleaned_sentence.append(mispell_dict[word])
        else:
            cleaned_sentence.append(word)
    return ' '.join(cleaned_sentence)
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
def sentences_cleaner(x):
  l=[]
  x=x.lower()
  x=x.split()
  for i in x:
    u =''.join(str(e) for e in i if e not in string.punctuation)
    u=u.strip()
    l.append(u)
  v =' '.join(str(e) for e in l)
  return v



df['english_sentence']=df['english_sentence'].apply(lambda x:re.sub("[^A-Za-z ]","",x))

df['english_sentence']=df['english_sentence'].apply(lambda x:deconcate(x))

df['english_sentence']=df["english_sentence"].apply(lambda x:sentences_cleaner(x))
```

```
df["english_sentence"]
```

```
0        another rs crore arrived in four instalments b...
1               youve got to just respond to the situation
2                       that the opposite may also be true
3                                    martial art movements
4                                             south africa
                              ...
39443    agra is a historic city and its evidences have...
39444      i am becoming a solution provider im very happy
39445    this glacier is km long km wide and almost met...
39446    and forgetting is the necessary result of any ...
39447    just like them all are brothers the sons of bh...
Name: english_sentence, Length: 39448, dtype: object
```

```
all_english_words=set()
for senten in df["english_sentence"]:
  for word in senten.split():
    all_english_words.add(word)
```

```
all_english_words
```

```
      'spanish',
      'occuring',
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
      'ashtadhyaya',
      'shut',
      'cryptotheme',
      'slowlv',
      'routes',
      'nuisance',
      'hermaphrodite',
      'kanungo',
      'regal',
      'meru',
      'segregation',
      'preconditions',
      'sides',
      'jehangir',
      'peonsof',
      'legislative',
      'contemplates',
      'remedying',
      'honouring',
      'eradicate',
      'halves',
      'allamaprabhu',
      'nisar',
```

```
 'inward',
 'womenthis',
 'hanged',
 'subbaraman',
 'bhartiya',
 'muscles',
 'loudspeaker',
 'sullivan',
 'writsordersinjunctions',
 'mumtobe',
 'dhan',
 'roll',
 'tai',
 'karanah',
 'malli',
 'thrown',
 'dharawi',
 'muavia',
 'rashiar',
 'bharat',
 'niyam',
 'everything',
 'propogated',
 'walled',
 'yourself',
 'financed',
 'mau',
 'dictum',
 'naming',
 ...}
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
38582
```

Hindi_Text_Preprocessing

```
# Converting The Data Into Lowercase
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x: x.lower())

# Removing English Words
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x:re.sub('[A-Za-z]',"",x))

# Removing Punch Symbols like '
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x: re.sub("'", '', x))

# Removing Special Characters
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x: ''.join(ch for ch in x if ch not in
```

```
# Removing Numberical values
remove_digits = str.maketrans('', '', digits)
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x: x.translate(remove_digits))


df['hindi_sentence'] = df['hindi_sentence'].apply(lambda x: re.sub("[२३०८१५७९४६]", "", x))



# Removing Extra empty spaces
df['hindi_sentence']=df['hindi_sentence'].apply(lambda x: x.strip())



# Adding The Start and End To Know the Starting and Ending of sentences
df['hindi_sentence'] = df['hindi_sentence'].apply(lambda x : 'START_ '+ x + ' _END')


all_hindi_words = set()
for sento in df["hindi_sentence"]:
  for word in sento.split():
    all_hindi_words.add(word)


len(all_hindi_words)
```

```
44333
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
'संस्तुत',
'मंत्रियों',
'आत्यांतिक',
'अल्पाहार',
'घुमावदार',
'मुलाकात',
'ओवल।',
'बृहस्पति',
'विषुव',
'पूर्वदृष्टांत',
'बालरूम',
'उन्हें”',
'ओप्पोस्सुम',
'सिखलाता',
'वियतनाम',
'भागमती',
'प्रहारों',
'हौलीवूड',
'छंबोबद्ध',
'मक्की',
'पाटिल',
'बोनस',
```

```
        'फामला',
        'महानासिका',
        'ःएअल्ट्ह्',
        'नुसरत',
        'कूपर',
        'खाई',
        'पियु',
        'डी',
        'सुदृढ़',
        'दशाओं',
        'पेर्पेत्रतोर्स',
        'तगडा',
        'अस्पतालों',
        'अर्धविक्षिप्त',
        'ज़ेन',
        'युगलों',
        'सकते',
        'मांगना',
        'डरार्नेधमकाने',
        'शैक्षक',
        'कारकीर्द',
        'थाजिसमें',
        'चालक',
        'वेटरों',
        'लगाओ',
        'हैवहां',
        'तनिकसा',
        'सोविएत',
        'कुरानी',
        'पली',
```



```
        'सोचेगें',
        ...}
```

Adding Extra Length Columns of Every Sentences

```
df["Length of English Sentences"]=df['english_sentence'].apply(lambda x:len(x.split(" ")))
```

```
df["Length of Hindi Sentences"]=df['hindi_sentence'].apply(lambda x:len(x.split(" ")))
```

```
df.head(3)
```

	english_sentence	hindi_sentence	Length of English Sentences	Length of Hindi Sentences
0	another rs crore arrived in	START_ पाए और मार्च और	11	17

```
df[df['Length of English Sentences']>30].shape
```

```
(3733, 4)
```

2	be true	उतनी ही वैध हो सक	7	12

```
df=df[df['Length of English Sentences']<=20]
df=df[df['Length of Hindi Sentences']<=20]
```

```
df.shape
```

```
(25767, 4)
```

```
df.head(3)
```

	english_sentence	hindi_sentence	Length of English Sentences	Length of Hindi Sentences
0	another rs crore arrived in four instalments b...	START_ पाए और मार्च और नवंबर के बीच चार किश्...	11	17
1				11
2	that the opposite may also be true	START_ बिल्कुल विपरीत सोच भी उतनी ही वैध हो सक...	7	12

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
df.tail(3)
```

	english_sentence	hindi_sentence	Length of English Sentences	Length of Hindi Sentences
39445	this glacier is km long km wide and almost met...	START_ यह हिमनद किमी लंबा व किमी चौड़ा और लग...	11	17
39446	and forgetting is the necessary result of any ...	START_ और इस प्रकार विस्मरण एक दीर्घकालकई शताब...	20	14
39447	just like them all are brothers the sons of bh...	START_ उन्हीं की तरह सभी भाईभाई हैं भारत माता...	11	14

Important Items

Max _Length :

```
max_length_source =max(df['Length of English Sentences'])
max_length_source
```

```
20
```

Maximum Length source [English] : 20

```
max_length_target = max(df['Length of Hindi Sentences'])
max_length_target
```

```
20
```

Maximum Length Target [Hindi] : 20

[English] Input Words In sorted order :::


```
'administrations',
'administrative',
'administratively',
'administrator',
'administrators',
'admirable',
'admiral',
'admiralty',
'admiration',
'admire',
'admired',
'admirer',
'admirers',
'admires',
'admiring',
'admissibility',
'admission',
'admissionwere',
'admit',
'admits',
'admittance',
'admitted',
'admittedly',
```

```
        'admitting',
        'admixture',
        'admonishes',
        'admonitions',
        'adobe',
        'adolescence',
        'adolescencies',
        'adolescents',
        'adolf',
        'adon',
        'adopt',
        'adopted',
        'adopting',
        'adoption',
        'adoptions',
        'adopts',
        'adorable',
        'adoration',
        'adore',
        'adores',
        'adorn',
        'adorned',
        'adornment',
        'adptations',
        'adra',
        'adressed',
        'adrift',
        'adrshclassic',
        'ads',
        'adtpanna',
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
        'adulterated',
        'adulteration',
```

[Hindi] Target Words In sorted order :::

```
target_words = sorted(list(all_hindi_words))
target_words
```

```
        'अध्याय।',
        'अध्यारोपित',
        'अध्यावरण',
        'अध्येता',
        'अध्ययन',
        'अध्वर्युगण',
        'अध्वर्युवर्ग',
        'अनंग',
        'अनंगपाल',
        'अनंत',
        'अनंतकाल',
        'अनंतपुर',
```

```
 'अनंतराम',
 'अनंतशायी',
 'अनंलकृत',
 'अनऋऊण्श्छष्तम',
 'अनऋऊण्श्छष्न',
 'अनऋऊण्श्छष्य',
 'अनऋऊण्श्छष्याय',
 'अनऋद्वय',
 'अनऋपचारिक',
 'अनऑफ़शियल',
 'अनगढ़',
 'अनगढ़े',
 'अनगिनत',
 'अनचाहा',
 'अनचाही',
 'अनचाहे',
 'अनच्छेद',
 'अनछुई',
 'अनजान',
 'अनजानसे',
 'अनजानीसी',
 'अनजाने',
 'अनत',
 'अनथक',
 'अनदेखा',
 'अनदेखी',
 'अनदेखे',
 'अनधिऋत',
 'अनधिक',
 'अनधिकृत',
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
 'अनन्तराम',
 'अनन्तशयनम',
 'अनन्य',
 'अनन्यभाव',
 'अनपढ',
 'अनपढ़',
 'अनपेक्षित',
 'अनप्रयोग',
 'अनबन',
 'अनबिकी',
 'अनबूझी',
 ...]
```

Tokens

```
num_encoder_tokens_E = len(input_words)
num_encoder_tokens_E
```

```
38582
```

```
num_decoder_tokens_H = len(target_words)
num_decoder_tokens_H
```

```
44333
```

```
num_decoder_tokens_H += 1 #for zero padding
```

```
num_encoder_tokens_E,num_decoder_tokens_H
```

```
(38582, 44333)
```

Encoder Tokens [English] : 12083

Decoder Tokens [Hindi] :14465

Indexing For Tokens

```
input_token_index = dict([(word,i+1) for i ,word in enumerate(input_words)])
```

```
target_token_index = dict([(word,i+1) for i,word in enumerate(target_words)])
```

```
ex))
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
, 'aachmanritual
```

Reversing Index

```
Rev_input_token_index = dict([(i+1,word) for i ,word in enumerate(input_words)])
```

```
Rev_target_token_index = dict([(i+1,word) for i,word in enumerate(target_words)])
```

```
print("Input : {} And Target : {}".format(Rev_input_token_index,Rev_target_token_index))
```

```
Input : {1: 'a', 2: 'aa', 3: 'aaa', 4: 'aaber', 5: 'aac', 6: 'aacharan', 7: 'aachmanritu
```

Shuffle

```
df = shuffle(df)
```

```
df.head(5)
```

	english_sentence	hindi_sentence	Length of English Sentences	Length of Hindi Sentences
22347	the problem is they are just too expensive ten...	START_ समस्या यह है कि वो बहुत महंगे है हज़ारो...	13	12
3359	in april this year the prime minister received...	START_ इस साल अप्रैल में प्रधानमंत्री को अपनी ...	17	17
20130	depends on one crucial fact	START_ एक अत्यंत महत्त्वपूर्ण बात पर निर्भर कर...	5	10
35589	similar answer	START_ सामान्य अन्तर _END	2	4
36510	with the help of dwanduwad islamic dharmashast...	START_ द्वंद्ववाद की मदद से इस्लामी धर्मशास्त्...	14	17

Train & Test Split Data

```
X,Y = df["english_sentence"],df['hindi_sentence']
```

```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random_state=30)
```



```
x_test.shape,y_test.shape
```

```
((7731,), (7731,))
```

Save The Data in Pickle Format

```
x_train.to_pickle('x_train.pkl')
x_test.to_pickle('x_test.pkl')
y_train.to_pickle('y_train.pkl')
y_test.to_pickle('y_test.pkl')
```

Generating The Batches For DataSet

Samples

```
np.zeros((128,20),dtype='float32')
```

```
array([[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=float32)
```

```
np.zeros((128,20,14446),dtype='float32')
```

```
array([[[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., 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=float32)
```

Let's Start's The Batches

```
def generate_batch(X =x_train,Y=y_train,batch_size=128):
  while True:
    for j in range(0,len(X),batch_size):

      #------------------     Layers Creation   ----------------------------
      encoder_input_data=np.zeros((batch_size,max_length_source),dtype='float32')
      #128 Rows & 20 Columns

      decoder_input_data = np.zeros((batch_size,max_length_target),dtype='float32')
      #128 Rows & 20 Columns

      decoder_target_data = np.zeros((batch_size,max_length_target,num_decoder_tokens_H),dtyp
      # It is Three Dimensional [128 Rows and 20 Columns] * 14446

      # first Iteration code logic i=0, [English Sentences]Good......., [Hindi Sentences]Aach
      for i,(input_text,target_text) in enumerate(zip(X[j:j+batch_size],Y[j:j+batch_size])):
        # Hers English Words Separated
        for t,word in enumerate(input_text.split()):
          encoder_input_data[i,t] = input_token_index[word] # encoder input seq
          #It is Example : For First Iteration encoder_input_data[0,0] =1256

        for t,word in enumerate(target_text.split()):
          # Above we created Decoder_Input_Data layer we are including that word index into l
          if  t<len(target_text.split()) -1:
            decoder_input_data[i,t] = target_token_index[word]
          if t>0:
            # Decoder Target like a decoder if word presented if represented 1 else 0
            # we are Keepig t>0 because we don't need START
            decoder_target_data[i,t-1,target_token_index[word]]=1
            # Above We Mention Three values ! Rember Decoder_Target_Data is 3 Dimensional
        yield([encoder_input_data,decoder_input_data],decoder_target_data)
```


Encoder & Decoder Architecture

Before Going Main Architecture Let Parctice Some Normal Things Ok!

```
data = np.array([[1,2,0,0]])
```

```
data.shape
#One Row And 4 Columns
```

```
(1, 4)
```

```
# Let's Build Small Example Model


x = Input(shape=(None,))
e = Embedding(5,5,mask_zero=True)(x)
m = Model(inputs=x,outputs=e)
p =m.predict(data)
```

```
print(p.shape)
```

```
(1, 4, 5)
```

```
print(p)
```

```
[[[ 0.01228404  0.00808271  0.04648527  0.00110035  0.02664003]
  [-0.01957023 -0.02899926  0.01669905 -0.04600201 -0.00611962]
  [-0.01468407  0.03634665 -0.04621065 -0.04093482 -0.0029062 ]
  [-0.01468407  0.03634665 -0.04621065 -0.04093482 -0.0029062 ]]]
```

Let's Start Buliding Encoder


```
# By Using Kera's we doing this
encoder_inputs = Input(shape=(None,))
encoder_embedding = Embedding(num_encoder_tokens_E,length_dimesnion,mask_zero=True)(encoder_i
encoder_lstm = LSTM(length_dimesnion,return_state=True)

encoder_outputs,hidden_state,cell_state = encoder_lstm(encoder_embedding)
# We Don't Need encoder_outputs present so we discard it
encoder_states = [hidden_state,cell_state]
```

Let's Start Buliding Decoder

```
decoder_inputs = Input(shape=(None,))
decoder_embedding = Embedding(num_decoder_tokens_H,length_dimesnion,mask_zero=True)
dec_emb = decoder_embedding(decoder_inputs)
decoder_lstm = LSTM(length_dimesnion,return_sequences=True,return_state=True)

#Here We don't need d_hidden_state,d_cell_state
decoder_outputs,d_hidden_state,d_cell_state = decoder_lstm(dec_emb,initial_state=encoder_stat

# Adding Dense Layer
```

```
# Adding Dense Layer
decoder_dense = Dense(num_decoder_tokens_H,activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

MODEL

```
model = Model([encoder_inputs,decoder_inputs],decoder_outputs)
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, None)]       0           []

 input_3 (InputLayer)           [(None, None)]       0           []

 embedding_1 (Embedding)        (None, None, 300)    11574600    ['input_2[0][0]']

 embedding_2 (Embedding)        (None, None, 300)    13299900    ['input_3[0][0]']

 lstm (LSTM)                    [(None, 300),        721200      ['embedding_1[0][0]']
                                 (None, 300),
                                 (None, 300)]

 lstm_1 (LSTM)                  [(None, None, 300),  721200      ['embedding_2[0][0]',
                                 (None, 300),                     'lstm[0][1]',
                                 (None, 300)]                     'lstm[0][2]']

 dense (Dense)                  (None, None, 44333)  13344233    ['lstm_1[0][0]']

==================================================================================================
Total params: 39,661,133
Trainable params: 39,661,133
Non-trainable params: 0
__________________________________________________________________________________________________
```


Training Model

```
train_samples = len(x_train)
test_samples = len(x_test)
batch_size = 128
epochs = 100
```

```
model.fit_generator(generator = generate_batch(x_train, y_train, batch_size = batch_size),
                    steps_per_epoch = train_samples//batch_size,
                    epochs=100,
                    validation_data = generate_batch(x_test, y_test, batch_size = batch_size)
                    validation_steps = test_samples//batch_size)
```

```
Epoch 73/100
140/140 [==============================] - 85s 605ms/step - loss: 0.2391 - accuracy:
Epoch 74/100
140/140 [==============================] - 85s 607ms/step - loss: 0.2198 - accuracy:
Epoch 75/100
140/140 [==============================] - 85s 610ms/step - loss: 0.2329 - accuracy:
Epoch 76/100
140/140 [==============================] - 85s 608ms/step - loss: 0.2223 - accuracy:
Epoch 77/100
140/140 [==============================] - 85s 606ms/step - loss: 0.2308 - accuracy:
Epoch 78/100
140/140 [==============================] - 85s 605ms/step - loss: 0.2342 - accuracy:
Epoch 79/100
140/140 [==============================] - 85s 609ms/step - loss: 0.2417 - accuracy:
Epoch 80/100
140/140 [==============================] - 85s 607ms/step - loss: 0.2564 - accuracy:
Epoch 81/100
140/140 [==============================] - 86s 612ms/step - loss: 0.2330 - accuracy:
Epoch 82/100
140/140 [==============================] - 85s 609ms/step - loss: 0.2356 - accuracy:
Epoch 83/100
140/140 [==============================] - 85s 606ms/step - loss: 0.2309 - accuracy:
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
6 - accuracy:
140/140 [==============================] - 85s 609ms/step - loss: 0.2455 - accuracy:
Epoch 86/100
140/140 [==============================] - 85s 608ms/step - loss: 0.2190 - accuracy:
Epoch 87/100
140/140 [==============================] - 85s 610ms/step - loss: 0.2563 - accuracy:
Epoch 88/100
140/140 [==============================] - 84s 603ms/step - loss: 0.2214 - accuracy:
Epoch 89/100
140/140 [==============================] - 85s 609ms/step - loss: 0.2372 - accuracy:
Epoch 90/100
140/140 [==============================] - 84s 604ms/step - loss: 0.2454 - accuracy:
Epoch 91/100
140/140 [==============================] - 85s 607ms/step - loss: 0.2461 - accuracy:
Epoch 92/100
140/140 [==============================] - 85s 605ms/step - loss: 0.2343 - accuracy:
Epoch 93/100
140/140 [==============================] - 84s 603ms/step - loss: 0.2406 - accuracy:
Epoch 94/100
140/140 [==============================] - 85s 609ms/step - loss: 0.2333 - accuracy:
Epoch 95/100
140/140 [==============================] - 84s 605ms/step - loss: 0.2403 - accuracy:
Epoch 96/100
140/140 [==============================] - 85s 611ms/step - loss: 0.2385 - accuracy:
```

```
Epoch 97/100
140/140 [==============================] - 86s 614ms/step - loss: 0.2348 - accuracy:
Epoch 98/100
140/140 [==============================] - 87s 619ms/step - loss: 0.2249 - accuracy:
Epoch 99/100
140/140 [==============================] - 85s 608ms/step - loss: 0.2262 - accuracy:
Epoch 100/100
140/140 [==============================] - 85s 607ms/step - loss: 0.2202 - accuracy:
<keras.callbacks.History at 0x7f7de033a5d0>
```

```
model.save_weights('weights.h5')


# Encoder Input Sequence thorugh the vectors
encoder_model = Model(encoder_inputs,encoder_states)


# Decoder
decoder_state_input_h = Input(shape = (length_dimesnion,))
decoder_state_input_c = Input(shape =(length_dimesnion,))

decoder_states_inputs =[decoder_state_input_h,decoder_state_input_c]

decoder_embedding2 = decoder_embedding(decoder_inputs)
decoder_outputs2, state_h2, state_c2 = decoder_lstm(decoder_embedding2,initial_state = decode
decoder_states2 = [state_h2, state_c2]
decoder_outputs2 = decoder_dense(decoder_outputs2) # Dense Softmax layer
```



```
decoder_Model = Model([decoder_inputs]+decoder_states_inputs,
                      [decoder_outputs2]+decoder_states2)


def decode_sequence(input_seq):
    # Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)
    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))
    # Populate the first character of target sequence with the start character.
    target_seq[0, 0] = target_token_index['START_']

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

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
```

```
        sampled_char = Rev_target_token_index[sampled_token_index]
        decoded_sentence += ' '+sampled_char

        # Exit condition: either hit max length
        # or find stop character.
        if (sampled_char == '_END' or
           len(decoded_sentence) > 50):
            stop_condition = True

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

        # Update states
        states_value = [h, c]

    return decoded_sentence


train_gen = generate_batch(x_train, y_train, batch_size = 1)
k=-1


k+=1
(input_seq, actual_output), _ = next(train_gen)
decoded_sentence = decode_sequence(input_seq)
print('Input English sentence:', x_train[k:k+1].values[0])
print('Actual Hindi Translation:', y_train[k:k+1].values[0][6:-4])
```



```
Input English sentence: which has simply pegged certain emotional rewards
Actual Hindi Translation:  हमने अपने आप को कुछ ऐसे भावनात्मक सुखों से जोड़ रखा है
Predicted Hindi Translation:  लेकिन कि इस पर अनुवाद हों तो जैसे


k+=1
(input_seq, actual_output), _ = next(train_gen)
decoded_sentence = decode_sequence(input_seq)
print('Input English sentence:', x_train[k:k+1].values[0])
print('Actual Hindi Translation:', y_train[k:k+1].values[0][6:-4])
print('Predicted Hindi Translation:', decoded_sentence[:-4])

Input English sentence: the other day i was in kerala my home state
Actual Hindi Translation:  कुछ दिनों पहले मैं केरल गया हुआ था मेरे गाँव
Predicted Hindi Translation:  लेकिन कुछ ऐसी अविश्वसनीय कि द्रव आपने आपने और व
```

✓ 0s completed at 12:59 PM

Automatic saving failed. This file was updated remotely or in another tab. Show diff