** In this Project our objective is to building a machine_translator ,Which can translate an English Sentence to Hindi **

** Here We will Implement a Custon Endocer_Dedocder(Seq-2-Seq) Model to archive our goal. **

In [ ]:
```
# importing Supporting Libraries
import pandas as pd
from tqdm.notebook import tqdm
from random import sample
import re
import numpy as np
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import pandas as pd
import re
import tensorflow as tf
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Model
import numpy as np
```

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

** Introduction of the dataset, We will use : **

We will use a Coupus published by IIT-Bombay. This corpus contains 1.6 Million santence pairs(Eng-Hindi) of both languages.The IIT Bombay English-Hindi corpus contains parallel corpus for English-Hindi as well as monolingual Hindi corpus collected from a variety of existing sources and corpora developed at the Center for Indian Language Technology, IIT Bombay over the years. This corpus has been used at the Workshop on Asian Language Translation Shared Task since 2016 the Hindi-to-English and English-to-Hindi languages pairs and as a pivot language pair for the Hindi-to-Japanese and Japanese-to-Hindi language pairs. To find more About the corpus ,Please go through the link given bellow : https://www.cfilt.iitb.ac.in/~parallelcorp/iitb_en_hi_parallel/ (https://www.cfilt.iitb.ac.in/~parallelcorp/iitb_en_hi_parallel/)

After Unziping the initial Corpus i get two folders :

```
1 - IITB.en-hi.en.txt (contains 1.6 million English Sentences.)
2 - IITB.en-hi.hi.txt (contains Hindi translation of each corresponding
 English Sentence.)
```

***** Reading the initial Text Files. *****

In [ ]:
```
english_corpus = open(r"/content/drive/MyDrive/1. My_folder/2. AI Projects./4. Ma
english_sentences = []
for line in english_corpus:
  english_sentences.append(line)
print("We have ",len(english_sentences),"English Sentences In our english_corpus.
```

```
We have  1609682 English Sentences In our english_corpus.
```

In [ ]:
```
hindi_corpus = open(r"/content/drive/MyDrive/1. My_folder/2. AI Projects./4. Mach
hindi_sentences = []
for line in hindi_corpus:
  hindi_sentences.append(line)
print("For Each English Sentence, We have corresponding",len(hindi_sentences),"Hi
```

```
For Each English Sentence, We have corresponding 1609682 Hindi Sentences In our
hindi_corpus.
```

***** Creating a Pandas Dataframe *****

In [ ]:
```
parell_dataset = pd.DataFrame()
parell_dataset["eng_sentences"] = english_sentences
parell_dataset["hin_sentences"] = hindi_sentences
parell_dataset.head(3)
```

Out[5]:

	eng_sentences	hin_sentences
0	Give your application an accessibility workout\n	अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें\n
1	Accerciser Accessibility Explorer\n	एक्सेर्साइसर पहुंचनीयता अन्वेषक\n
2	The default plugin layout for the bottom panel\n	निचले पटल के लिए डिफोल्ट प्लग-इन खाका\n

***** Taking 5 lac pairs of sentences Randomly. And Creating a small dataset,Which we will use to train our model. *****

In [ ]:
```
indexex_of_all_points = list(range(len(parell_dataset)))
subset_of_indexes = sample(indexex_of_all_points, 500000)
hindi_eng_dataset = pd.DataFrame()
for index in tqdm(subset_of_indexes):
  hindi_eng_dataset = hindi_eng_dataset.append(parell_dataset.iloc[index])
print("Now We have ",len(hindi_eng_dataset),"pair of english-Hindi sentences in o
print("-"*60)
hindi_eng_dataset.head()
```

```
HBox(children=(FloatProgress(value=0.0, max=500000.0), HTML(value='')))


Now We have  500000 pair of english-Hindi sentences in our dataframe.
------------------------------------------------------------
```

Out[10]:

	eng_sentences	hin_sentences
393227	Announce to the hypocrites that they shall hav...	मुनाफ़िको (कपटाचारियों) को मंगल-सूचना दे दो कि...
183336	Responsible for the nice application SVG Icon.\n	अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्तरदायी. \n
1579176	First meeting of Task Force was held on 4th De...	टास्क फोर्स की पहली बैठक 4 दिसंबर 2017 को हुई ...
860509	shenanigan\n	नटखटपन\n
322020	But if they wax proud (and persist in their at...	लेकिन यदि वे घमंड करें (और अल्लाह को याद न करे...

***** Saving Dataset in Memory. *****

In [ ]:
```
hindi_eng_dataset.to_csv("hindi_eng_dataset.csv",index = False)
```

In [ ]:
```
!cp "/content/hindi_eng_dataset.csv" "/content/drive/MyDrive/1. My_folder/2. AI P
```

***** Reading the dataset for Preprocessing and some Data analysis. *****

In [ ]:
```
hindi_eng_dataset = pd.read_csv(r"/content/drive/MyDrive/1. My_folder/2. AI Proje
print("Now We have ",len(hindi_eng_dataset),"pair of english-Hindi sentences in o
print("-"*60)
hindi_eng_dataset.head()
```

```
Now We have  1000 pair of english-Hindi sentences in our dataframe.
------------------------------------------------------------
```

Out[3]:

	eng_sentences	hin_sentences
0	Announce to the hypocrites that they shall hav...	मुनाफ़िको (कपटाचारियों) को मंगल-सूचना दे दो कि...
1	Responsible for the nice application SVG Icon.\n	अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्तरदायी. \n
2	First meeting of Task Force was held on 4th De...	टास्क फोर्स की पहली बैठक 4 दिसंबर 2017 को हुई ...
3	shenanigan\n	नटखटपन\n
4	But if they wax proud (and persist in their at...	लेकिन यदि वे घमंड करें (और अल्लाह को याद न करे...

In [ ]:
```
hindi_eng_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 2 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   eng_sentences  500000 non-null  object
 1   hin_sentences  500000 non-null  object
dtypes: object(2)
memory usage: 7.6+ MB
```

In [ ]:
```
def decontractions(phrase):
    """decontracted takes text and convert contractions into natural form.
     ref: https://stackoverflow.com/questions/19790188/expanding-english-language
    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(text):
    text = text.lower()
    text = decontractions(text)
    text = re.sub('[^A-Za-z0-9 ]+', '', text)
    return text

def preprocess_hin(text):
    text = text.lower()
    text = decontractions(text)
    text = text.strip('\n')
    return text
```

In [ ]:
```
hindi_eng_dataset['eng_sentences'] = hindi_eng_dataset['eng_sentences'].apply(pre
hindi_eng_dataset['hin_sentences'] = hindi_eng_dataset['hin_sentences'].apply(pre
hindi_eng_dataset.head()
```

Out[49]:

	eng_sentences	hin_sentences
0	announce to the hypocrites that they shall hav...	मुनाफ़िको (कपटाचारियों) को मंगल-सूचना दे दो कि...
1	responsible for the nice application svg icon	अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्तरदायी.
2	first meeting of task force was held on 4th de...	टास्क फोर्स की पहली बैठक 4 दिसंबर 2017 को हुई थी।
3	shenanigan	नटखटपन
4	but if they wax proud and persist in their att...	लेकिन यदि वे घमंड करें (और अल्लाह को याद न करे...

In [ ]:
```
lengths_of_english_sentences = [len(sentence.split()) for sentence in hindi_eng_d
print("50th percentile Value of all lengths of headline is :",np.percentile(lengt
print("90th percentile Value of all lengths of headline is :",np.percentile(lengt
print("95th percentile Value of all lengths of headline is :",np.percentile(lengt
print("99th percentile Value of all lengths of headline is :",np.percentile(lengt
print("99.9th percentile Value of all lengths of headline is :",np.percentile(len
print("99.99th percentile Value of all lengths of headline is :",np.percentile(le
print("100th percentile Value of all lengths of headline is :",np.percentile(leng
```

```
50th percentile Value of all lengths of headline is : 8.0
90th percentile Value of all lengths of headline is : 30.0
95th percentile Value of all lengths of headline is : 39.0
99th percentile Value of all lengths of headline is : 68.0
99.9th percentile Value of all lengths of headline is : 129.0
99.99th percentile Value of all lengths of headline is : 196.0
100th percentile Value of all lengths of headline is : 653.0
```

In [ ]:
```
lengths_of_hindi_sentences = [len(sentence.split()) for sentence in hindi_eng_dat
print("50th percentile Value of all lengths of headline is :",np.percentile(lengt
print("90th percentile Value of all lengths of headline is :",np.percentile(lengt
print("95th percentile Value of all lengths of headline is :",np.percentile(lengt
print("99th percentile Value of all lengths of headline is :",np.percentile(lengt
print("99.9th percentile Value of all lengths of headline is :",np.percentile(len
print("99.99th percentile Value of all lengths of headline is :",np.percentile(le
print("100th percentile Value of all lengths of headline is :",np.percentile(leng
```

```
50th percentile Value of all lengths of headline is : 9.0
90th percentile Value of all lengths of headline is : 33.0
95th percentile Value of all lengths of headline is : 44.0
99th percentile Value of all lengths of headline is : 77.0
99.9th percentile Value of all lengths of headline is : 142.0010000000475
99.99th percentile Value of all lengths of headline is : 212.0
100th percentile Value of all lengths of headline is : 695.0
```

***** Transforming data in a specific form, which my model takes : *****

In [ ]:
```
hindi_eng_dataset['hindi_inp'] = '<start> ' + hindi_eng_dataset['hin_sentences'].
hindi_eng_dataset['hindi_out'] = hindi_eng_dataset['hin_sentences'].astype(str) +
hindi_eng_dataset = hindi_eng_dataset.drop(['hin_sentences'], axis=1)
hindi_eng_dataset.head(4)
```

Out[52]:

	eng_sentences	hindi_inp	hindi_out
0	announce to the hypocrites that they shall hav...	<start> मुनाफ़िको (कपटाचारियों) को मंगल-सूचना ...	मुनाफ़िको (कपटाचारियों) को मंगल-सूचना दे दो कि...
1	responsible for the nice application svg icon	<start> अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्...	अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्तरदायी. ...
2	first meeting of task force was held on 4th de...	<start> टास्क फोर्स की पहली बैठक 4 दिसंबर 2017...	टास्क फोर्स की पहली बैठक 4 दिसंबर 2017 को हुई ...
3	shenanigan	<start> नटखटपन	नटखटपन <end>

In [ ]:
```
print("One sample English Sentence is :\n",hindi_eng_dataset.loc[23565]["eng_sent
print("-"*60)
print("Corresponding Hindi Sentence for input is :\n",hindi_eng_dataset.loc[23565
print("-"*60)
print("Corresponding Hindi Sentence for output is :\n",hindi_eng_dataset.loc[2356
```

```
One sample English Sentence is :
 he concluded his thesis with the following words
------------------------------------------------------------
Corresponding Hindi Sentence for input is :
 <start> अपना प्रबन्ध उनहोंने निम्नांकित शब्दों में समाप्त किया:
------------------------------------------------------------
Corresponding Hindi Sentence for output is :
 अपना प्रबन्ध उनहोंने निम्नांकित शब्दों में समाप्त किया:  <end>
```

In [ ]:
```
hindi_eng_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 3 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   eng_sentences   500000 non-null  object
 1   hindi_inp       500000 non-null  object
 2   hindi_out       500000 non-null  object
dtypes: object(3)
memory usage: 11.4+ MB
```

***** Saving Dataset in my Memory. *****

In [ ]:
```
hindi_eng_dataset.to_csv("hindi_eng_dataset_preprocessed_sample.csv",index= False
```

In [ ]:
```
!cp "/content/hindi_eng_dataset_preprocessed_sample.csv" "/content/drive/MyDrive/
```

***** Loading Basic Preprocessed Data, also reducing the size of dataset because traing model with 5 Lac points was becoming hard : *****

In [3]:
```
import pandas as pd
hindi_eng_dataset_preprocessed_sample = pd.read_csv("/content/drive/MyDrive/Machi
print("Shape of data is :",hindi_eng_dataset_preprocessed_sample.shape)
print("-"*60)
hindi_eng_dataset_preprocessed_sample.head(4)
```

```
Shape of data is : (250000, 3)
------------------------------------------------------------
```

Out[3]:

	eng_sentences	hindi_inp	hindi_out
0	announce to the hypocrites that they shall hav...	<start> मुनाफ़िको (कपटाचारियों) को मंगल-सूचना ...	मुनाफ़िको (कपटाचारियों) को मंगल-सूचना दे दो कि...
1	responsible for the nice application svg icon	<start> अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्...	अच्छे अनुप्रयोग एसवीजी प्रतीक हेतु उत्तरदायी. ...
2	first meeting of task force was held on 4th de...	<start> टास्क फोर्स की पहली बैठक 4 दिसंबर 2017...	टास्क फोर्स की पहली बैठक 4 दिसंबर 2017 को हुई ...
3	shenanigan	<start> नटखटपन	नटखटपन <end>

***** Checking null Values,And dropping those rows. *****

In [4]:
```
hindi_eng_dataset_preprocessed_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250000 entries, 0 to 249999
Data columns (total 3 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   eng_sentences  249853 non-null  object
 1   hindi_inp      250000 non-null  object
 2   hindi_out      250000 non-null  object
dtypes: object(3)
memory usage: 5.7+ MB
```

In [5]:
```
hindi_eng_dataset_preprocessed_sample.isnull().sum(axis = 0)
```

Out[5]:
```
eng_sentences    147
hindi_inp          0
hindi_out          0
dtype: int64
```

In [6]:
```
hindi_eng_dataset_preprocessed_sample  = hindi_eng_dataset_preprocessed_sample.dr
hindi_eng_dataset_preprocessed_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 249853 entries, 0 to 249999
Data columns (total 3 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   eng_sentences  249853 non-null  object
 1   hindi_inp      249853 non-null  object
 2   hindi_out      249853 non-null  object
dtypes: object(3)
memory usage: 7.6+ MB
```

In [7]:
```
hindi_eng_dataset_preprocessed_sample.isnull().sum(axis = 0)
```

Out[7]:
```
eng_sentences    0
hindi_inp        0
hindi_out        0
dtype: int64
```

***** Splitting the dataset into train and Validation *****

In [8]:
```
from sklearn.model_selection import train_test_split
train_pairs, validation_pairs = train_test_split(hindi_eng_dataset_preprocessed_s
print("Shape of train_pairs  is :",train_pairs.shape)
print("Shape of train_pairs  is :",validation_pairs.shape)
```

```
Shape of train_pairs  is : (212375, 3)
Shape of train_pairs  is : (37478, 3)
```

In [9]:
```
train_pairs.head(3)
```

Out[9]:

	eng_sentences	hindi_inp	hindi_out
78267	ice cream	<start> आइस्क्रीम	आइस्क्रीम <end>
17623	enabling font s	<start> प्रविष्टि सक्षम किया जा रहा है...	प्रविष्टि सक्षम किया जा रहा है... <end>
28236	and lo he is a witness unto that	<start> और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है	और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है <end>

In [10]:
```
validation_pairs.head(3)
```

Out[10]:

	eng_sentences	hindi_inp	hindi_out
105838	brasero optical media library	<start> ब्रैसेरो ऑप्टिकल मीडिया लाइब्रेरी	ब्रैसेरो ऑप्टिकल मीडिया लाइब्रेरी <end>
136008	in the one place you multiply it by 5311 i e t...	<start> एक जगह आप उसे 5311 से अर्थात उस संख्या...	एक जगह आप उसे 5311 से अर्थात उस संख्या से गुण...
193275	the condition of operations which is working w...	<start> कार्य संचालन की ऐसी स्थिति जब कार्य अच...	कार्य संचालन की ऐसी स्थिति जब कार्य अच्छी तरह ...

In [11]:
```
# for one sentence we will be adding <end> token so that the tokanizer learns the
# with this we can use only one tokenizer for both encoder output and decoder out
train_pairs.iloc[0]['hindi_inp']= str(train_pairs.iloc[0]['hindi_inp'])+' <end>'
train_pairs.iloc[0]['hindi_out']= str(train_pairs.iloc[0]['hindi_inp'])+' <end>'
train_pairs.head(3)
```

Out[11]:

	eng_sentences	hindi_inp	hindi_out
78267	ice cream	<start> आइस्क्रीम <end>	<start> आइस्क्रीम <end> <end>
17623	enabling font s	<start> प्रविष्टि सक्षम किया जा रहा है...	प्रविष्टि सक्षम किया जा रहा है... <end>
28236	and lo he is a witness unto that	<start> और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है	और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है <end>

In [12]:
```
print("train_pairs Data Head :")
print("-"*100)
train_pairs.head(3)
```

```
train_pairs Data Head :
--------------------------------------------------------------------------------
--------------------
```

Out[12]:

	eng_sentences	hindi_inp	hindi_out
78267	ice cream	<start> आइस्क्रीम <end>	<start> आइस्क्रीम <end> <end>
17623	enabling font s	<start> प्रविष्टि सक्षम किया जा रहा है...	प्रविष्टि सक्षम किया जा रहा है... <end>
28236	and lo he is a witness unto that	<start> और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है	और यक़ीनी ख़ुदा भी उससे वाक़िफ़ है <end>

In [13]:
```
print("validation_pairs Data Head :")
print("-"*100)
validation_pairs.head(3)
```

```
validation_pairs Data Head :
--------------------------------------------------------------------------------
--------------------
```

Out[13]:

	eng_sentences	hindi_inp	hindi_out
105838	brasero optical media library	<start> ब्रैसेरो ऑप्टिकल मीडिया लाइबेरी	ब्रैसेरो ऑप्टिकल मीडिया लाइबेरी <end>
136008	in the one place you multiply it by 5311 i e t...	<start> एक जगह आप उसे 5311 से अर्थात उस संख्या...	एक जगह आप उसे 5311 से अर्थात उस संख्या से गुण...
193275	the condition of operations which is working w...	<start> कार्य संचालन की ऐसी स्थिति जब कार्य अच...	कार्य संचालन की ऐसी स्थिति जब कार्य अच्छी तरह ...

In [14]:
```
hindi_eng_dataset_preprocessed_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 249853 entries, 0 to 249999
Data columns (total 3 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   eng_sentences  249853 non-null  object
 1   hindi_inp      249853 non-null  object
 2   hindi_out      249853 non-null  object
dtypes: object(3)
memory usage: 7.6+ MB
```

In [15]:
```
hindi_eng_dataset_preprocessed_sample.isnull().sum(axis = 0)
```

Out[15]:
```
eng_sentences    0
hindi_inp        0
hindi_out        0
dtype: int64
```

***** Creating Tokenizers, Doing Padding,truncating,coverting sentences into numerical IDs *****

In [16]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [17]:
```
tknizer_eng = Tokenizer()
tknizer_eng.fit_on_texts(train_pairs['eng_sentences'].values)
encoder_seq = tknizer_eng.texts_to_sequences(train_pairs['eng_sentences'].values)
max_len_eng = 40
padded_english_train = pad_sequences(encoder_seq, maxlen=max_len_eng, dtype='int3
```

In [18]:
```
# For validation data
encoder_seq = tknizer_eng.texts_to_sequences(validation_pairs['eng_sentences'].va
padded_english_validation = pad_sequences(encoder_seq, maxlen=max_len_eng, dtype=
```

In [19]:
```
padded_english_train[0]
```

Out[19]:
```
array([3240, 8221,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    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 [20]:
```
padded_english_validation[0]
```

Out[20]:
```
array([3643, 6655, 1168, 1449,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    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 [21]:
```
tknizer_hindi = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n')
tknizer_hindi.fit_on_texts(train_pairs['hindi_inp'].values)
decoder_inp_seq = tknizer_hindi.texts_to_sequences(train_pairs['hindi_inp'].value
max_len_hindi = 45
padded_input_hindi = pad_sequences(decoder_inp_seq, maxlen=max_len_hindi, dtype=
```

In [22]:
```
# For validation data
seq = tknizer_hindi.texts_to_sequences(validation_pairs['hindi_inp'].values)
val_padded_input_hindi = pad_sequences(seq, maxlen=max_len_hindi, dtype='int32',
```

In [23]:
```
padded_input_hindi[0]
```

Out[23]:
```
array([    1, 58050, 58051,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     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 [24]:
```
val_padded_input_hindi[0]
```

Out[24]:
```
array([    1,  4172,  8457,  1634, 23142,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     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 [25]:
```
# For Decoder_output
decoder_out_seq = tknizer_hindi.texts_to_sequences(train_pairs['hindi_out'].value
padded_output_hindi = pad_sequences(decoder_out_seq, maxlen=max_len_hindi, dtype=
```

In [26]:
```
# For validation data
seq = tknizer_hindi.texts_to_sequences(validation_pairs['hindi_out'].values)
val_padded_output_hindi = pad_sequences(seq, maxlen=max_len_hindi, dtype='int32',
```

In [27]:
```
padded_output_hindi[0]
```

Out[27]:
```
array([    1, 58050, 58051, 58051,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     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 [28]:
```
val_padded_output_hindi[0]
```

Out[28]:
```
array([ 4172,  8457,  1634, 23142, 58051,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     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 [29]:
```
vocab_size_english=len(tknizer_eng.word_index.keys())+1
print("Vocab size of English Sentences is :",vocab_size_english)
vocab_size_hindi=len(tknizer_hindi.word_index.keys())+1
print("-"*100)
print("Vocab size of hindi Sentences is :",vocab_size_hindi)
```

```
Vocab size of English Sentences is : 83653
--------------------------------------------------------------------------------
--------------------
Vocab size of hindi Sentences is : 130674
```

In [30]:
```
hindi_index_to_word={}
for key,value in tknizer_hindi.word_index.items():
    hindi_index_to_word[value]=key
print(len(hindi_index_to_word))
```

```
130673
```

** Creating Custom Encoder,Decoder Models: **

In [32]:
```
import pandas as pd
import re
import tensorflow as tf
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Model
import numpy as np
```

** Defining Encoder **

In [33]:
```
class Encoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns encoder-outputs,
    encoder_final_state_h,encoder_final_state_c

    '''

    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        self.lstm_output = 0
        self.lstm_state_h=0
        self.lstm_state_c=0
        self.lstm_size = lstm_size
        #Initialize Embedding layer
        self.embedding = Embedding(input_dim = inp_vocab_size, output_dim = embed
                                   input_length = input_length,
                                   mask_zero=True, name="embedding_layer_encoder"
        #Intialize Encoder LSTM layer
        self.lstm = LSTM(lstm_size, return_state=True, return_sequences=True, nam


    def call(self,input_sequence,states):

        '''
          This function takes a sequence input and the initial states of the enco
          Pass the input_sequence input to the Embedding layer,
          Pass the embedding layer ouput to encoder_lstm
          returns -- encoder_output, last time step's hidden and cell state
        '''

        input_embedd = self.embedding(input_sequence)

        self.lstm_output, self.lstm_state_h,self.lstm_state_c = self.lstm(input_e

        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 zeros is of size [32,lstm_units]
      '''
      return (tf.zeros([batch_size, self.lstm_size]),
              tf.zeros([batch_size, self.lstm_size]))
```

** Defining Decoder **

In [34]:
```
class Decoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns output sequence
    '''

    def __init__(self,out_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        self.vocab_size = out_vocab_size
        self.embedding_dim = embedding_size
        self.lstm_size = lstm_size
        self.input_length = input_length

        #Initialize Embedding layer
        self.embedding = Embedding(input_dim=self.vocab_size, output_dim=self.emb
                                   input_length=self.input_length,
                                   mask_zero=True, name="embedding_layer_decoder"

        #Intialize Decoder LSTM layer
        self.lstm = LSTM(self.lstm_size, return_sequences=True, return_state=True


    def call(self,input_sequence,initial_states):
        '''
          This function takes a sequence input and the initial states of the enco
          Pass the input_sequence input to the Embedding layer,
          Pass the embedding layer ouput to decoder_lstm

          returns -- decoder_output,decoder_final_state_h,decoder_final_state_c
        '''
        target_embedd                                = self.embedding(input_sequence)
        lstm_output, decoder_h,decoder_c             = self.lstm(target_embedd, initia

        return lstm_output,decoder_h,decoder_c
```

** Combining Both (Encoder model & Decoder Model) **

In [35]:
```
class Encoder_decoder(tf.keras.Model):

    def __init__(self,encoder_inputs_length,decoder_inputs_length,output_vocab_si
                 vocab_size_eng,vocab_size_hindi):

        super().__init__()
        self.vocab_size_eng = vocab_size_eng
        self.encoder_inputs_length = encoder_inputs_length
        self.vocab_size_eng = vocab_size_hindi
        self.decoder_inputs_length = decoder_inputs_length
        self.output_vocab_size = output_vocab_size
        #Create encoder object
        self.encoder = Encoder(inp_vocab_size=self.vocab_size_eng, embedding_size
                               lstm_size = 256 ,
                               input_length=self.encoder_inputs_length)


        #Create decoder object
        self.decoder = Decoder(out_vocab_size=self.vocab_size_hindi ,
                               embedding_size=100, lstm_size = 256 ,
                               input_length=self.decoder_inputs_length)


        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        self.dense   = Dense(self.output_vocab_size, activation='softmax')


    def call(self,data):
        input,output = data[0], data[1]
        encoder_output, encoder_h, encoder_c = self.encoder(input,0)
        states = [encoder_h, encoder_c]
        decoder_output ,decoder_h,decoder_c  = self.decoder(output, states)
        output                               = self.dense(decoder_output)
        return output
```

** Defining Encoder_Decoder Model **

In [36]:
```
#Create an object of encoder_decoder Model class,
model  = Encoder_decoder(encoder_inputs_length=40,decoder_inputs_length=45,
                         output_vocab_size=vocab_size_hindi,
                         vocab_size_ita = vocab_size_english,vocab_size_eng= voca
```

** Compiling Model **

In [37]:
```
optimizer = tf.keras.optimizers.Adam()
model.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy')
```

In [91]:
```
model.summary()
```

```
Model: "encoder_decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
encoder (Encoder)            multiple                  8730868   
_________________________________________________________________
decoder (Decoder)            multiple                  13432968  
_________________________________________________________________
dense (Dense)                multiple                  33583218  
=================================================================
Total params: 55,747,054
Trainable params: 55,747,054
Non-trainable params: 0
_________________________________________________________________
```

** Training Whole network in multiple steps **

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1534s 458ms/step - loss: 2.0203 -
val_loss: 1.6983
Epoch 2/2
3319/3319 [==============================] - 1539s 464ms/step - loss: 1.6504 -
val_loss: 1.5312
```

Out[40]:
```
<tensorflow.python.keras.callbacks.History at 0x7fde9953ded0>
```

In [ ]:
```
model.save_weights("model_weights_after_2_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_2_epochs.h5" "/content/drive/MyDrive/Machine_translatior
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1517s 457ms/step - loss: 1.4573 -
val_loss: 1.4459
Epoch 2/2
3319/3319 [==============================] - 1519s 458ms/step - loss: 1.3122 -
val_loss: 1.3971
```

Out[49]:
```
<tensorflow.python.keras.callbacks.History at 0x7fde408f4850>
```

In [ ]:
```
model.save_weights("model_weights_after_4_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_4_epochs.h5" "/content/drive/MyDrive/Machine_translatior
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1517s 457ms/step - loss: 1.1937 -
val_loss: 1.3667
Epoch 2/2
3319/3319 [==============================] - 1514s 456ms/step - loss: 1.0932 -
val_loss: 1.3515
```

Out[52]:
```
<tensorflow.python.keras.callbacks.History at 0x7fde3883e390>
```

In [ ]:
```
model.save_weights("model_weights_after_6_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_6_epochs.h5" "/content/drive/MyDrive/Machine_translatior
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1529s 461ms/step - loss: 1.0067 -
val_loss: 1.3460
Epoch 2/2
3319/3319 [==============================] - 1517s 457ms/step - loss: 0.9326 -
val_loss: 1.3481
```

Out[55]:
```
<tensorflow.python.keras.callbacks.History at 0x7fde41131650>
```

In [ ]:
```
model.save_weights("model_weights_after_8_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_8_epochs.h5" "/content/drive/MyDrive/Machine_translatior
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1516s 457ms/step - loss: 0.8705 -
val_loss: 1.3507
Epoch 2/2
3319/3319 [==============================] - 1517s 457ms/step - loss: 0.8178 -
val_loss: 1.3598
```

Out[58]:
```
<tensorflow.python.keras.callbacks.History at 0x7fde99ab9a10>
```

In [ ]:
```
model.save_weights("model_weights_after_10_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_10_epochs.h5" "/content/drive/MyDrive/Machine_translatic
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1461s 440ms/step - loss: 0.7957 -
val_loss: 1.3630
Epoch 2/2
3319/3319 [==============================] - 1431s 431ms/step - loss: 0.7382 -
val_loss: 1.3755
```

Out[39]:
```
<tensorflow.python.keras.callbacks.History at 0x7f13b39be1d0>
```

In [ ]:
```
model.save_weights("model_weights_after_12_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_12_epochs.h5" "/content/drive/MyDrive/1. My_folder/2. AI
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1430s 431ms/step - loss: 0.7014 -
val_loss: 1.3905
Epoch 2/2
3319/3319 [==============================] - 1445s 435ms/step - loss: 0.6694 -
val_loss: 1.4064
```

Out[42]:
```
<tensorflow.python.keras.callbacks.History at 0x7f13b30c7a50>
```

In [ ]:
```
model.save_weights("model_weights_after_12_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_12_epochs.h5" "/content/drive/MyDrive/1. My_folder/2. AI
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1449s 437ms/step - loss: 0.6411 -
val_loss: 1.4224
Epoch 2/2
3319/3319 [==============================] - 1446s 436ms/step - loss: 0.6156 -
val_loss: 1.4399
```

Out[49]:
```
<tensorflow.python.keras.callbacks.History at 0x7f13b31753d0>
```

In [ ]:
```
model.save_weights("model_weights_after_14_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_14_epochs.h5" "/content/drive/MyDrive/1. My_folder/2. AI
```

In [ ]:
```
model.fit([padded_english_train, padded_input_hindi], padded_output_hindi ,
          epochs = 2,
          validation_data = ([padded_english_validation,val_padded_input_hindi],v
          verbose = True,
          batch_size = 64)
```

```
Epoch 1/2
3319/3319 [==============================] - 1445s 435ms/step - loss: 0.5926 -
val_loss: 1.4600
Epoch 2/2
3319/3319 [==============================] - 1441s 434ms/step - loss: 0.5720 -
val_loss: 1.4780
```

Out[52]:
```
<tensorflow.python.keras.callbacks.History at 0x7f13b29c5ad0>
```

In [ ]:
```
model.save_weights("model_weights_after_16_epochs.h5")
```

In [ ]:
```
!cp "model_weights_after_16_epochs.h5" "/content/drive/MyDrive/1. My_folder/2. AI
```

** Combing Whole data piple line in one function "translate_it_to_hindi" **

In [61]:
```
def translate_it_to_hindi(model, input_sentence):
    predicted_word =''
    output_sentence =''
    seq    = tknizer_eng.texts_to_sequences([input_sentence])
    tokens = pad_sequences(seq, maxlen=40, dtype='int32', padding='post')
    initial_state = model.layers[0].initialize_states(1024)
    encoder_outputs, final_state_h, final_state_c = model.layers[0](tokens,initia
    input  = np.array([[1]],dtype=np.int32)
    states = [final_state_h,final_state_c]
    while(predicted_word!='<end>'):
        decoder_output,decoder_state_h,decoder_state_c = model.layers[1](input, i
        output = model.layers[2](decoder_output)
        states = [decoder_state_h,decoder_state_c]
        output_word_index = np.argmax(output[0],axis=1)
        #print(output_word_index)
        #predicted_word = tknizer_hindi.index_word[output_word_index[0]]
        predicted_word = tknizer_hindi.index_word[output_word_index[0]]
        input = tknizer_hindi.word_index[predicted_word]
        input = np.array([[input]],dtype=np.int32)
        if (predicted_word!='<end>'):
            output_sentence+=predicted_word+" "
        else:
            output_sentence+=predicted_word
    return output_sentence
```

***** Translating some sentences by our model from the dataset itself :*****

In [77]:
```
hindi_eng_dataset = pd.read_csv(r"/content/drive/MyDrive/Machine_translation_Proj
data_point = hindi_eng_dataset.iloc[0]
english_sentence = str(data_point["eng_sentences"])
original_hindi_sentence = data_point["hindi_out"]
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Origianl original_hindi_sentence Sentece is :",original_hindi_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model)
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : announce to the hypocrites that they sh
all have a painful chastisement
--------------------------------------------------------------------------------
----------------------------------------
Origianl original_hindi_sentence Sentece is : मुनाफ़िको (कपटाचारियों) को मंगल-सूचना
दे दो कि उनके लिए दुखद यातना है;  <end>
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : मुनाफ़िको कपटाचारियों को मंगल सूचना
दे दो कि उनके लिए दुखद यातना है <end>
```

In [78]:
```
data_point = hindi_eng_dataset.iloc[100]
english_sentence = str(data_point["eng_sentences"])
original_hindi_sentence = data_point["hindi_out"]
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Origianl original_hindi_sentence Sentece is :",original_hindi_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : primitively
--------------------------------------------------------------------------------
----------------------------------------
Origianl original_hindi_sentence Sentece is : मूलतः <end>
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : मूलतः <end>
```

In [84]:
```
data_point = hindi_eng_dataset.iloc[500]
english_sentence = str(data_point["eng_sentences"])
original_hindi_sentence = data_point["hindi_out"]
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Origianl original_hindi_sentence Sentece is :",original_hindi_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : housing scheme for scheduled castes and
denotified tribes external website that opens in a new window
--------------------------------------------------------------------------------
----------------------------------------
Origianl original_hindi_sentence Sentece is : अनुसूचित जातियों और अधिसूचित जनजातियों
के लिए आवास योजना (बाहरी वेबसाइट जो एक नई विंडों में खुलती हैं)  <end>
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : अनुसूचित जातियों और अनुसूचित जन
जातियों के लिए आवास क्षेत्र एवं आवास केन्द्र द्वारा गठित एक समिति <end>
```

***** Translating some general sentences :*****

In [85]:
```python
english_sentence = str("I love my Mom.")
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model)
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : I love my Mom.
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : मैं माँ को प्यार चाहता हूँ। <end>
```

In [86]:
```python
english_sentence = str("it is very popular.")
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model)
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : it is popular.
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : काफ़ी लोकप्रिय है। <end>
```

In [87]:
```python
english_sentence = str("India is a big country")
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model)
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : India is a big country
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : भारत देश का एक बड़ा देश है। <end>
```

In [90]:
```python
english_sentence = str("today is my day")
translated_sen_by_model = translate_it_to_hindi(model,english_sentence)
print("-"*120)
print("Original english_sentence Sentence is :",english_sentence)
print("-"*120)
print("Translated translated_sen_by_model By Model is :",translated_sen_by_model)
```

```
--------------------------------------------------------------------------------
----------------------------------------
Original english_sentence Sentence is : today is my day
--------------------------------------------------------------------------------
----------------------------------------
Translated translated_sen_by_model By Model is : आज आज सुबह <end>
```

** Thank You..!! :) **