**Naan Mudhalvan**

**Artificial Intelligence**

**Phase - 3**

Create a chatbot in python

**Importing required libraries :**

import tensorflow as tf

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from tensorflow.keras.layers import TextVectorization

import re,string

from tensorflow.keras.layers import LSTM,Dense,Embedding,Dropout,LayerNormalization

**Reading Dataset :**

df=pd.read\_csv('/kaggle/input/simple-dialogs-for chatbot/dialogs.txt',sep='\t', names=['question','answer'])

print(f'Dataframe size: {len(df)}')

df.head()

**Data Preprocessing :**

* **Data Visualization :**

df['question tokens']=df['question'].apply(lambda x:len(x.split()))

df['answer tokens']=df['answer'].apply(lambda x:len(x.split()))

plt.style.use('fivethirtyeight')

fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))

sns.set\_palette('Set2')

sns.histplot(x=df['question tokens'],data=df,kde=True,ax=ax[0])

sns.histplot(x=df['answer tokens'],data=df,kde=True,ax=ax[1])

sns.jointplot(x='question tokens',y='answer tokens',data=df,kind='kde',fill=True,cmap='YlGnBu')

plt.show()

* **Text Cleaning :**

def clean\_text(text):

text=re.sub('-',' ',text.lower())

text=re.sub('[.]',' . ',text)

text=re.sub('[1]',' 1 ',text)

text=re.sub('[2]',' 2 ',text)

text=re.sub('[3]',' 3 ',text)

text=re.sub('[4]',' 4 ',text)

text=re.sub('[5]',' 5 ',text)

text=re.sub('[6]',' 6 ',text)

text=re.sub('[7]',' 7 ',text)

text=re.sub('[8]',' 8 ',text)

text=re.sub('[9]',' 9 ',text)

text=re.sub('[0]',' 0 ',text)

text=re.sub('[,]',' , ',text)

text=re.sub('[?]',' ? ',text)

text=re.sub('[!]',' ! ',text)

text=re.sub('[$]',' $ ',text)

text=re.sub('[&]',' & ',text)

text=re.sub('[/]',' / ',text)

text=re.sub('[:]',' : ',text)

text=re.sub('[;]',' ; ',text)

text=re.sub('[\*]',' \* ',text)

text=re.sub('[\']',' \' ',text)

text=re.sub('[\"]',' \" ',text)

text=re.sub('\t',' ',text)

return text

df.drop(columns=['answer tokens','question tokens'],axis=1,inplace=True)

df['encoder\_inputs']=df['question'].apply(clean\_text)

df['decoder\_targets']=df['answer'].apply(clean\_text)+' <end>'

df['decoder\_inputs']='<start> '+df['answer'].apply(clean\_text)+' <end>'

df.head(10)

df['encoder input tokens']=df['encoder\_inputs'].apply(lambda x:len(x.split()))

df['decoder input tokens']=df['decoder\_inputs'].apply(lambda x:len(x.split()))

df['decoder target tokens']=df['decoder\_targets'].apply(lambda x:len(x.split()))

plt.style.use('fivethirtyeight')

fig,ax=plt.subplots(nrows=1,ncols=3,figsize=(20,5))

sns.set\_palette('Set2')

sns.histplot(x=df['encoder input tokens'],data=df,kde=True,ax=ax[0])

sns.histplot(x=df['decoder input tokens'],data=df,kde=True,ax=ax[1])

sns.histplot(x=df['decoder target tokens'],data=df,kde=True,ax=ax[2])

sns.jointplot(x='encoder input tokens',y='decoder target tokens',data=df,kind='kde',fill=True,cmap='YlGnBu')

plt.show()

print(f"After preprocessing: {' '.join(df[df['encoder input tokens'].max()==df['encoder input tokens']]['encoder\_inputs'].values.tolist())}")

print(f"Max encoder input length: {df['encoder input tokens'].max()}")

print(f"Max decoder input length: {df['decoder input tokens'].max()}")

print(f"Max decoder target length: {df['decoder target tokens'].max()}")

df.drop(columns=['question','answer','encoder input tokens','decoder input tokens','decoder target tokens'],axis=1,inplace=True)

params={

"vocab\_size":2500,

"max\_sequence\_length":30,

"learning\_rate":0.008,

"batch\_size":149,

"lstm\_cells":256,

"embedding\_dim":256,

"buffer\_size":10000

}

learning\_rate=params['learning\_rate']

batch\_size=params['batch\_size']

embedding\_dim=params['embedding\_dim']

lstm\_cells=params['lstm\_cells']

vocab\_size=params['vocab\_size']

buffer\_size=params['buffer\_size']

max\_sequence\_length=params['max\_sequence\_length']

df.head(10)

* **Tokenization :**

vectorize\_layer=TextVectorization(

max\_tokens=vocab\_size,

standardize=None,

output\_mode='int',

output\_sequence\_length=max\_sequence\_length

)

vectorize\_layer.adapt(df['encoder\_inputs']+' '+df['decoder\_targets']+' <start> <end>')

vocab\_size=len(vectorize\_layer.get\_vocabulary())

print(f'Vocab size: {len(vectorize\_layer.get\_vocabulary())}')

print(f'{vectorize\_layer.get\_vocabulary()[:12]}')

def sequences2ids(sequence):

return vectorize\_layer(sequence)

def ids2sequences(ids):

decode=''

if type(ids)==int:

ids=[ids]

for id in ids:

decode+=vectorize\_layer.get\_vocabulary()[id]+' '

return decode

x=sequences2ids(df['encoder\_inputs'])

yd=sequences2ids(df['decoder\_inputs'])

y=sequences2ids(df['decoder\_targets'])

print(f'Question sentence: hi , how are you ?')

print(f'Question to tokens: {sequences2ids("hi , how are you ?")[:10]}')

print(f'Encoder input shape: {x.shape}')

print(f'Decoder input shape: {yd.shape}')

print(f'Decoder target shape: {y.shape}')

print(f'Encoder input: {x[0][:12]} ...')

print(f'Decoder input: {yd[0][:12]} ...') # shifted by one time step of the target as input to decoder is the output of the previous timestep

print(f'Decoder target: {y[0][:12]} ...')

data=tf.data.Dataset.from\_tensor\_slices((x,yd,y))

data=data.shuffle(buffer\_size)

train\_data=data.take(int(.9\*len(data)))

train\_data=train\_data.cache()

train\_data=train\_data.shuffle(buffer\_size)

train\_data=train\_data.batch(batch\_size)

train\_data=train\_data.prefetch(tf.data.AUTOTUNE)

train\_data\_iterator=train\_data.as\_numpy\_iterator()

val\_data=data.skip(int(.9\*len(data))).take(int(.1\*len(data)))

val\_data=val\_data.batch(batch\_size)

val\_data=val\_data.prefetch(tf.data.AUTOTUNE)

\_=train\_data\_iterator.next()

print(f'Number of train batches: {len(train\_data)}')

print(f'Number of training data: {len(train\_data)\*batch\_size}')

print(f'Number of validation batches: {len(val\_data)}')

print(f'Number of validation data: {len(val\_data)\*batch\_size}')

print(f'Encoder Input shape (with batches): {\_[0].shape}')

print(f'Decoder Input shape (with batches): {\_[1].shape}')

print(f'Target Output shape (with batches): {\_[2].shape}')

In this document to create a chatbot in python, the process began with the import of essential libraries such as TensorFlow, NumPy, Pandas, Matplotlib, and Seaborn, followed by the reading of a dataset. Data preprocessing involved data visualization through histograms and joint plots, offering insights into question and answer token lengths. Text cleaning was executed using regular expressions to manage symbols and punctuation. The dataset was modified to include encoder and decoder inputs and targets. Tokenization was achieved using the TextVectorization layer, enabling sequence-to-ID conversion for data preparation. The data was shuffled and split into training and validation sets, with batch configurations and prefetching optimized for efficient data processing. Overall, the project laid a solid foundation for the development of a functional chatbot, encompassing critical data preprocessing stages and necessary data structuring for subsequent model development and training.