

# CREATE A CHATBOT IN PYTHON

## Phase 3: Development part 1

Start building the chatbot model by loading and preprocessing the dataset.

Coding:

#import the required libraries

```
In [1]: 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
```

## # Load the dataset

```
In [2]: 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()
```

Dataframe size: 3725

Output:

Out[2]:

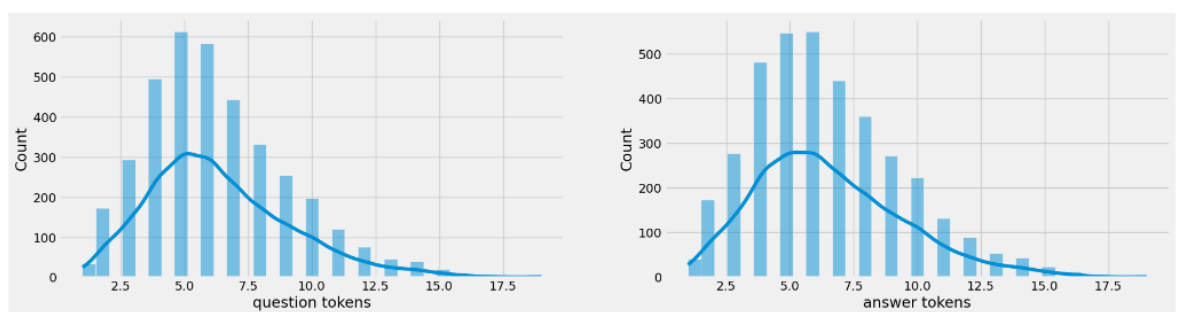
	question	answer
0	hi, how are you doing?	i'm fine. how about yourself?
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
2	i'm pretty good. thanks for asking.	no problem. so how have you been?
3	no problem. so how have you been?	i've been great. what about you?
4	i've been great. what about you?	i've been good. i'm in school right now.

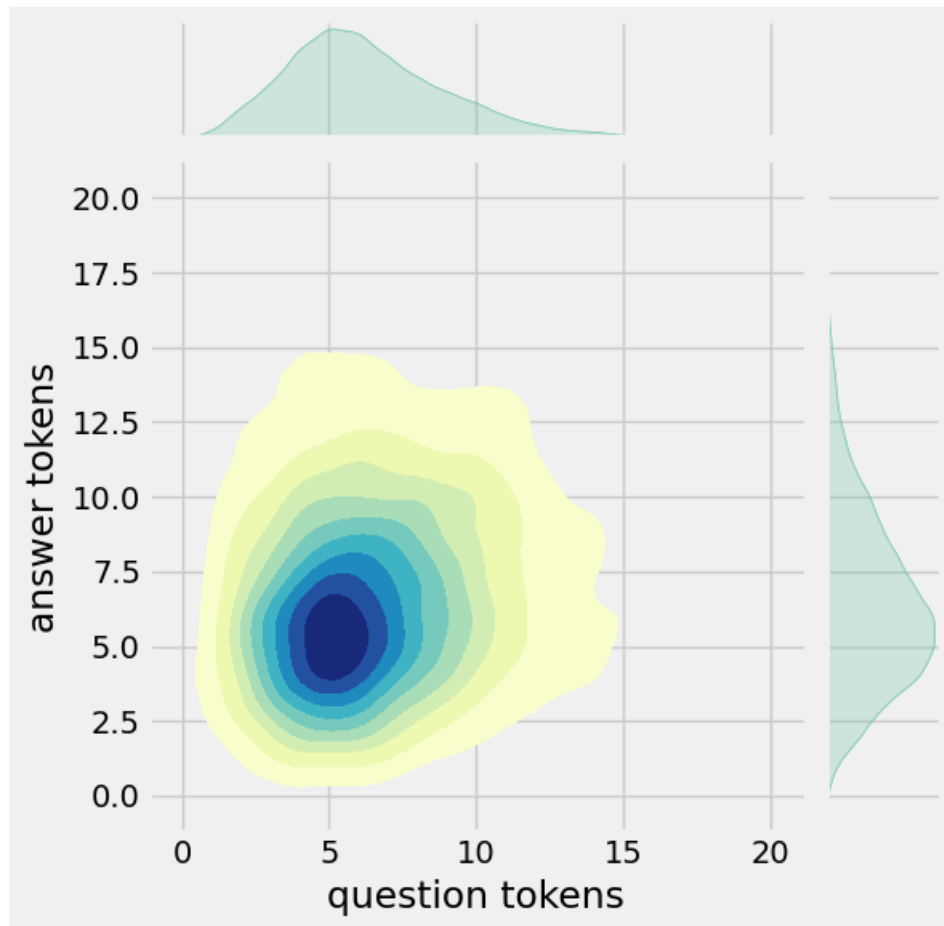
## Data Preprocessing

- Data Visualization

```
In [3]: 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()
```

Output:





- Text Cleaning

```
In [4]: 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('[\"']',' \" ',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)

```

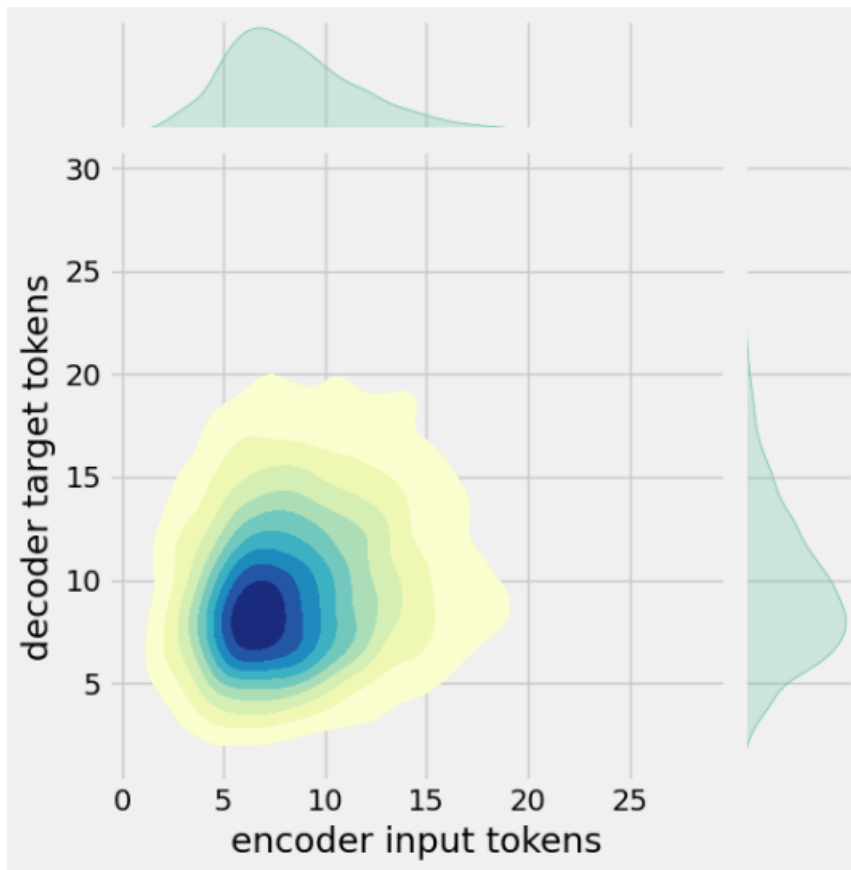
## Output:

Out[4]:

	question	answer	encoder_inputs	decoder_targets	decoder_inputs
0	hi, how are you doing?	i'm fine. how about yourself?	hi , how are you doing ?	i ' m fine . how about yourself ? <end>	<start> i ' m fine . how about yourself ? <end>
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.	i ' m fine . how about yourself ?	i ' m pretty good . thanks for asking . <end>	<start> i ' m pretty good . thanks for asking...
2	i'm pretty good. thanks for asking.	no problem. so how have you been?	i ' m pretty good . thanks for asking .	no problem . so how have you been ? <end>	<start> no problem . so how have you been ? ...
3	no problem. so how have you been?	i've been great. what about you?	no problem . so how have you been ?	i ' ve been great . what about you ? <end>	<start> i ' ve been great . what about you ? ...
4	i've been great. what about you?	i've been good. i'm in school right now.	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...	<start> i ' ve been good . i ' m in school ri...
5	i've been good. i'm in school right now.	what school do you go to?	i ' ve been good . i ' m in school right now .	what school do you go to ? <end>	<start> what school do you go to ? <end>
6	what school do you go to?	i go to pcc.	what school do you go to ?	i go to pcc . <end>	<start> i go to pcc . <end>
7	i go to pcc.	do you like it there?	i go to pcc .	do you like it there ? <end>	<start> do you like it there ? <end>
8	do you like it there?	it's okay. it's a really big campus.	do you like it there ?	it ' s okay . it ' s a really big campus . <...>	<start> it ' s okay . it ' s a really big cam...
9	it's okay. it's a really big campus.	good luck with school.	it ' s okay . it ' s a really big campus .	good luck with school . <end>	<start> good luck with school . <end>

```
In [5]: 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,cm
ap='YlGnBu')
plt.show()
```

Output:



In [6]:

```
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)
```

## Output:

Out[6]:

	encoder_inputs	decoder_targets	decoder_inputs
0	hi , how are you doing ?	i ' m fine . how about yourself ? <end>	<start> i ' m fine . how about yourself ? <end>
1	i ' m fine . how about yourself ?	i ' m pretty good . thanks for asking . <end>	<start> i ' m pretty good . thanks for asking...
2	i ' m pretty good . thanks for asking .	no problem . so how have you been ? <end>	<start> no problem . so how have you been ? ...
3	no problem . so how have you been ?	i ' ve been great . what about you ? <end>	<start> i ' ve been great . what about you ? ...
4	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...	<start> i ' ve been good . i ' m in school ri...
5	i ' ve been good . i ' m in school right now .	what school do you go to ? <end>	<start> what school do you go to ? <end>
6	what school do you go to ?	i go to pcc . <end>	<start> i go to pcc . <end>
7	i go to pcc .	do you like it there ? <end>	<start> do you like it there ? <end>
8	do you like it there ?	it ' s okay . it ' s a really big campus . <...>	<start> it ' s okay . it ' s a really big cam...
9	it ' s okay . it ' s a really big campus .	good luck with school . <end>	<start> good luck with school . <end>

# Tokenization

```
In [7]: 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]}')
```

Vocab size: 2443

['', '[UNK]', '<end>', '.', '<start>', '"', 'i', '?', 'you', ',', 'the', 'to']

```
In [10]: 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}')
```

Question sentence: hi , how are you ?

Question to tokens: [1971 9 45 24 8 7 0 0 0 0]

Encoder input shape: (3725, 30)

Decoder input shape: (3725, 30)

Decoder target shape: (3725, 30)

In [9]:

```
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]} ...')
```

```
Encoder input: [1971    9   45   24    8  194    7    0    0    0    0    0] ...
Decoder input: [   4    6    5  38 646    3  45  41 563    7    2    0] ...
Decoder target: [   6    5  38 646    3  45  41 563    7    2    0    0] ...
```

In [10]:

```
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}')
```

Number of train batches: 23

Number of training data: 3427

Number of validation batches: 3

Number of validation data: 447

Encoder Input shape (with batches): (149, 30)

Decoder Input shape (with batches): (149, 30)

Target Output shape (with batches): (149, 30)