ChatBot in Python

Let us finally get to building the project!

We will do this in 2 parts

1. Setup a Web framework
2. Preprocess the given data in Python

Web Framework(Flask)

Since the chatbot is the main objective here let us use a simple but flexible framework such as Flask. It is a micro web framework written in Python. It is classified as a micro-framework because it does not require any external tools or libraries to develop web applications. Flask is designed to be easy to use and flexible, and it can be used to develop a wide variety of web applications, from simple websites to complex web applications[1].

Instaling Flask

To install Flask for web development, you can follow these steps:

1. **Install Python.** Flask is a Python framework, so we will need to have Python installed on our computer. We can download Python from the Python website or install using a package manager.
   Since we are using macbook let us use brew

```
brew install python
```

2. **Install a virtual environment.** A virtual environment is a Python environment that is isolated from our system environment. This allows us to install packages without affecting other Python projects. To install a virtual environment, we can use the following command:

```
pip install virtualenv
```

3. **Create a virtual environment.** Once we have installed virtualenv, we can create a virtual environment for our Flask project using the following command:

```
virtualenv AI-Chatbot
```

4. **Activate the virtual environment.** Once we have created a virtual environment, we need to activate it before we can install Flask. To activate the virtual environment, you can use the following command:

```
source AI-Chatbot/bin/activate
```

5. **Install Flask.** Once we have activated the virtual environment, we can install Flask using the following command:

```
pip install Flask
```

6. **Verify that Flask is installed correctly.** To verify that Flask is installed correctly, we can try the following command:

```
python -m flask
```

```
Usage: python -m flask [OPTIONS] COMMAND [ARGS]...

  A general utility script for Flask applications.

  An application to load must be given with the '--app' option,
'FLASK_APP'
  environment variable, or with a 'wsgi.py' or 'app.py' file in the
current
  directory.

Options:
  -e, --env-file FILE   Load environment variables from this file.
python-
                        dotenv must be installed.
  -A, --app IMPORT      The Flask application or factory function to
load, in
```

```
                        the form 'module:name'. Module can be a dotted
import
                        or file path. Name is not required if it is
'app',
                        'application', 'create_app', or 'make_app',
and can be
                        'name(args)' to pass arguments.
  --debug / --no-debug  Set debug mode.
  --version             Show the Flask version.
  --help                Show this message and exit.

Commands:
  routes  Show the routes for the app.
  run     Run a development server.
  shell   Run a shell in the app context.
```

As we can see the flask was properly installed.

Trial Run Flask

To use Flask to develop the back-end of a web application, we will need to create a Flask application object. We can then use the Flask application object to register routes, which are the paths that users can request to access different pages of our web application.
Once we have registered routes, we can use the Flask application object to generate responses to requests. The responses can be HTML, CSS, JavaScript, or any other type of data that we want to send to the user.

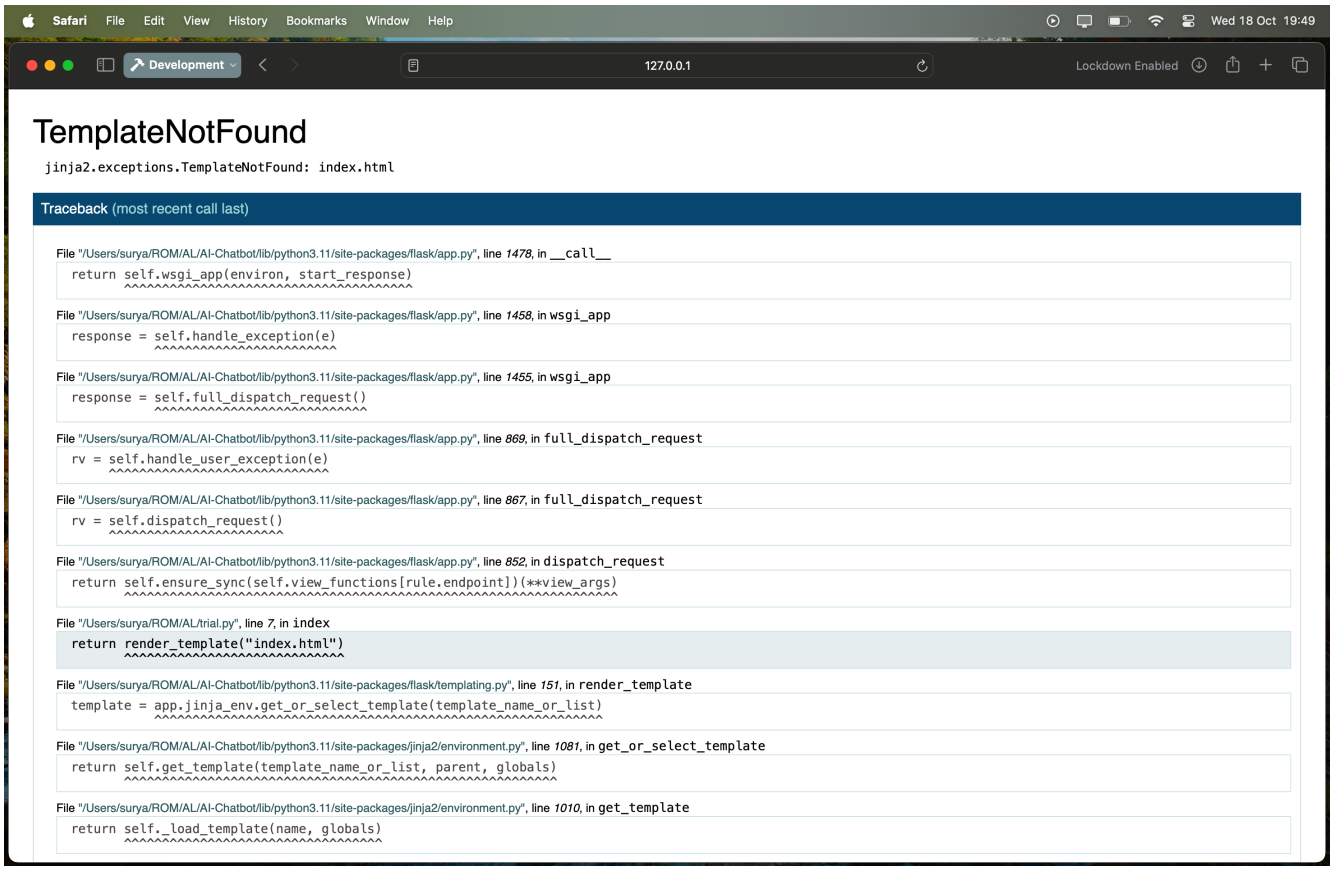
```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")

if __name__ == "__main__":
    app.run(debug=True)
```

```
 * Serving Flask app 'trial'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 142-167-126
```



As shown in above we are able to access Flask Application.

Preprocessing Dataset

Getting the Libraries

Installing the Libraries

```
pip install numpy
pip install matplotlib
```

```
pip install seaborn
pip install tensorflow
```

Importing the Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.layers import TextVectorization
import re,string
from tensorflow.keras.layers import
LSTM,Dense,Embedding,Dropout,LayerNormalization
```

Importing the Dataset

```
df=pd.read_csv('dialogs.txt',sep='\t',names=['Qustion','Answer'])
print(f'Dataset size: {len(df)}')
df.head()
```

```
Dataset size: 3725
```

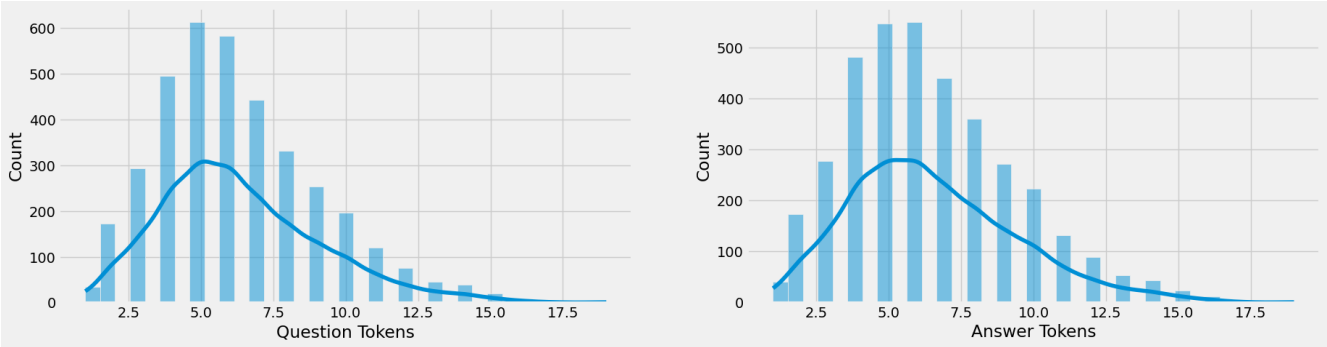
	Question	Answer
0	hi, how are you doing?	i'm fine. how about yourself?
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
2	i'm pretty good. thanks for asking.	no problem. so how have you been?
3	no problem. so how have you been?	i've been great. what about you?
4	i've been great. what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

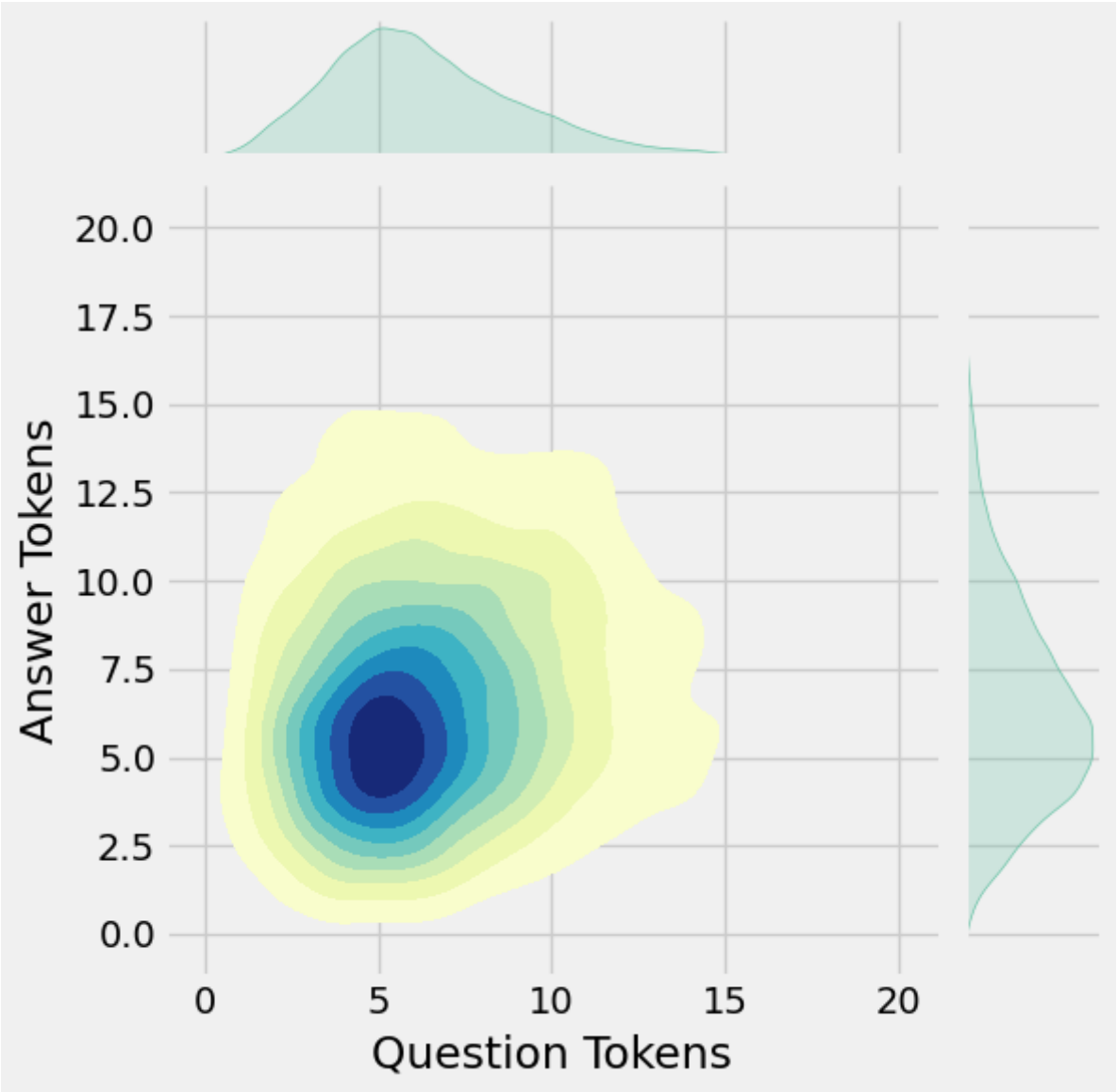
Data Preprocessing

To Process the Data properly we first need to understand the data first, So Let us first visualize the given Dataset first.
We will mainly use Seaborn for this.

Data Visualization

```
df['Qustion Tokens']=df['Qustion'].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['Qustion Tokens'],data=df,kde=True,ax=ax[0])
sns.histplot(x=df['Answer Tokens'],data=df,kde=True,ax=ax[1])
sns.jointplot(x='Qustion Tokens',y='Answer
Tokens',data=df,kind='kde',fill=True,cmap='YlGnBu')
plt.show()
```





Data Cleaning

Now that we are having a idea about the Dataset Let us Proceed towards Cleaning the data

```
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)
```

Qustion	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 ?
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 .
2	i'm pretty good.	no problem. so how have you been?	i ' m pretty good . thanks for asking .	no problem . so how have you been ?

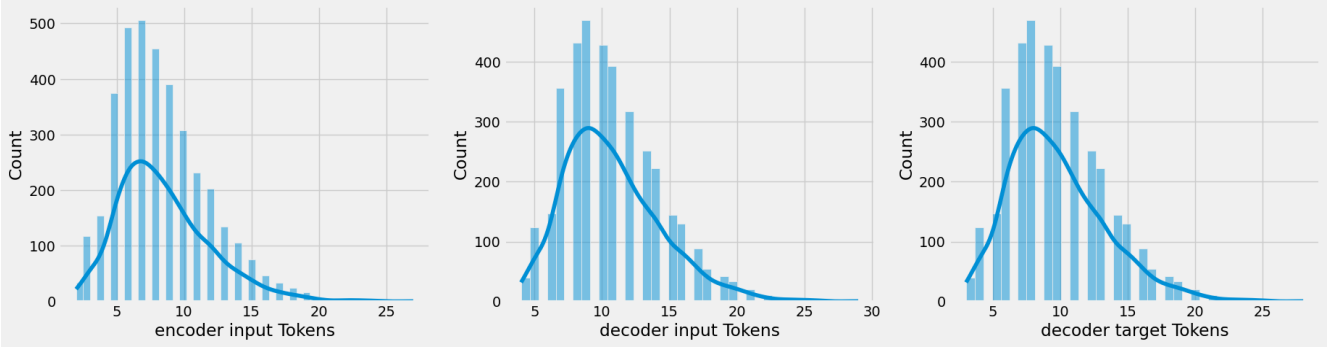
Qustion	Answer	encoder_inputs	decoder_targets	decoder_inputs
	thanks for asking.			
3	no problem. so how have you been?	i've been great. what about you?	no problem . so how have you been ?	i ' ve been great . what about you ?
4	i've been great. what about you?	i've been good. i'm in school right now.	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...
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 ?
6	what school do you go to?	i go to pcc.	what school do you go to ?	i go to pcc .
7	i go to pcc.	do you like it there?	i go to pcc .	do you like it there ?
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 . <...
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 .

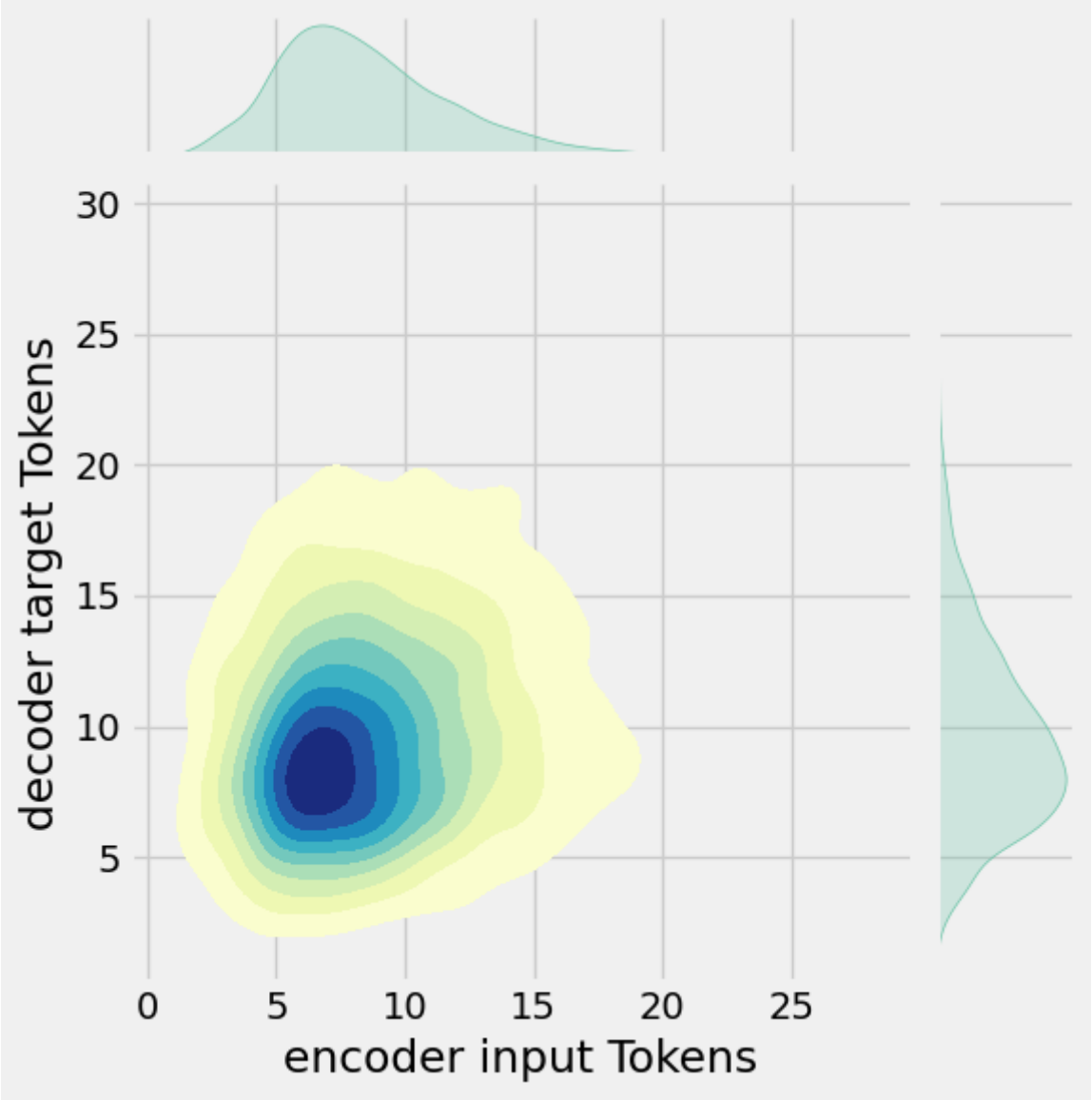
Now that we have cleaned the Dataset, Let us vizualize again

Re:Data Visualization

```
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)
```

```
After preprocessing: for example ,  if your birth date is january  1
2  ,   1  9  8  7  ,  write  0  1  /  1  2  /  8  7  .
Max encoder input length: 27
Max decoder input length: 29
Max decoder target length: 28
```

	encoder_inputs	decoder_targets
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 ?

	encoder_inputs	decoder_targets
4	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...
5	i ' ve been good . i ' m in school right now .	what school do you go to ?
6	what school do you go to ?	i go to pcc .
7	i go to pcc .	do you like it there ?
8	do you like it there ?	it ' s okay . it ' s a really big campus . <...
9	it ' s okay . it ' s a really big campus .	good luck with school .

Tokenization

Now let us Tokenize our dataset

```
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']
```

```
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}')
```

```
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)
```

```
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] ...
```

```
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)
```

With this we have successfully Preprocessed the Dataset

---

1. https://www.tutorialfor.com/questions-105322.html↩