# IBM PROJECT AI101

ARTIFICIAL INTELLIGENCE-GROUP 3

PROJECT – 3. CHAT BOT

Creating a chatbot for exceptional customer service is a complex task, but I can provide you with a high-level outline of the steps involved in designing and developing such a chatbot in Python.

1. **Functionality: Define the Scope**
    - Determine the specific tasks and questions your chatbot will handle. For example, it could answer frequently asked questions, provide product recommendations, assist with troubleshooting, or handle booking requests.
    - Create a list of intents or user queries that the chatbot should be able to recognize and respond to.

2. **User Interface: Design the Interface**
    - Decide where you want to integrate the chatbot (website, mobile app, or both).
    - Design a user-friendly and intuitive interface for users to interact with the chatbot. This could be a chatbox, a voice-based interface, or a combination of both.
    - Ensure that the interface is responsive and works well on various devices and screen sizes.

3. **Natural Language Processing (NLP): Implement NLP Techniques**
    - Choose an NLP library or framework for Python, such as spaCy, NLTK, or Hugging Face Transformers.
    - Preprocess user input to clean and tokenize the text.
    - Train or fine-tune an NLP model on your dataset to understand user intent and extract relevant information.
    - Implement techniques for entity recognition to identify important details in user queries.

4. **Responses: Plan Bot Responses**
    - Create a database of responses or templates that the chatbot can use to generate answers.
    - Implement a response generation system that selects the most appropriate response based on the user's query and the context.
    - Ensure that responses are concise, accurate, and user-friendly.

5. **Integration: Integrate with Website/App**
    - Depending on your platform, integrate the chatbot using appropriate libraries or frameworks. For example, you can use JavaScript for web integration or mobile app development tools for mobile apps.
    - Set up communication between the chatbot and your backend server if necessary.
    - Implement a user authentication system if personalization is required.

6. **Testing and Improvement: Continuous Enhancement**
    - Conduct thorough testing of the chatbot to identify and fix any issues or limitations.
    - Gather user feedback and monitor user interactions to identify areas for improvement.
    - Continuously update the chatbot's responses and functionality based on user feedback and changing business requirements.

7. **Dataset and Training**
    - Use the provided dataset as a starting point for training your chatbot.
    - Preprocess the dataset, clean the text, and structure it into intents and responses.
    - Train or fine-tune your NLP model using this dataset to improve the chatbot's understanding of user queries.

8. **Deployment and Maintenance**
    - Deploy your chatbot to a production environment and monitor its performance.
    - Implement logging and analytics to track user interactions and diagnose issues.
    - Regularly update and maintain the chatbot to keep it up-to-date with new information, products, or services.

Remember that building an exceptional customer service chatbot is an iterative process. You will likely need to refine and expand its capabilities over time based on user feedback and evolving business needs. Additionally, you may consider integrating machine learning techniques to improve the chatbot's performance and personalization.

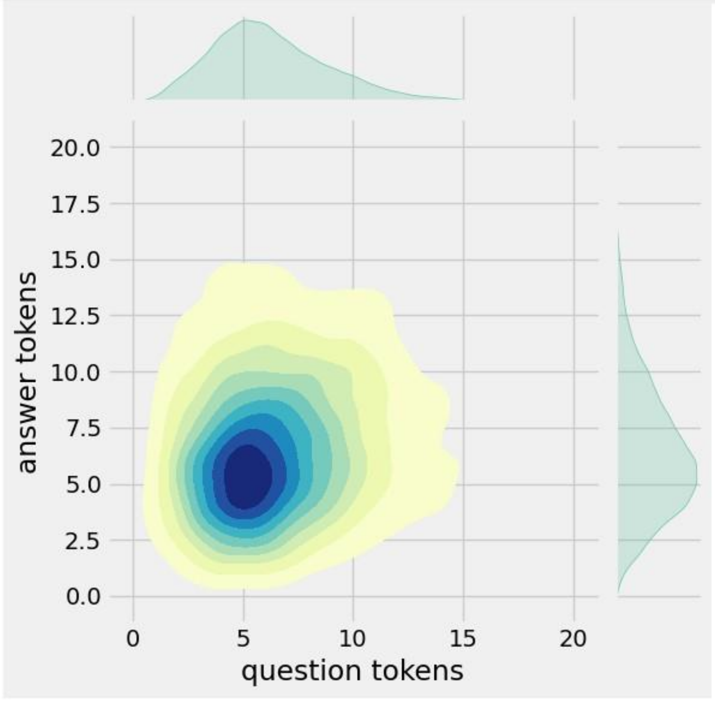
## PROJECT CODE:

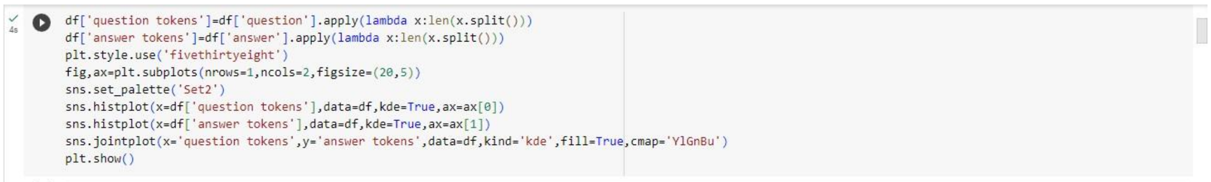
```python
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.layers import TextVectorization
import re,string
from tensorflow.keras.layers import LSTM,Dense,Embedding,Dropout,LayerNormalization
```

```python
df=pd.read_csv('dialogs.txt',sep='\t',names=['question','answer'])
print(f'Dataframe size: {len(df)}')
df.head()
```

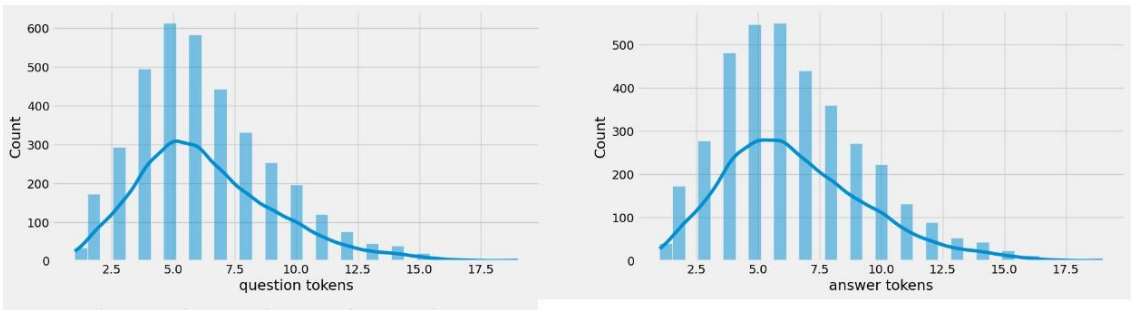
```
Dataframe size: 3725
```

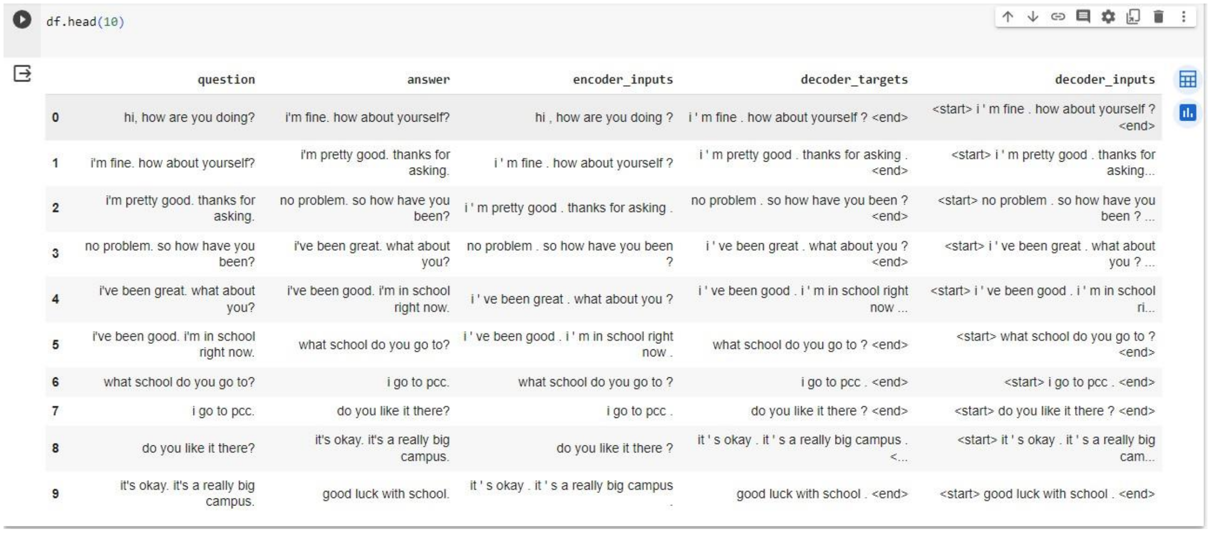
| | question | answer |
|---|---|---|
| 0 | hi, how are you doing? | i'm fine. how about yourself? |
| 1 | i'm fine. how about yourself? | i'm pretty good. thanks for asking. |
| 2 | i'm pretty good. thanks for asking. | no problem. so how have you been? |
| 3 | no problem. so how have you been? | i've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





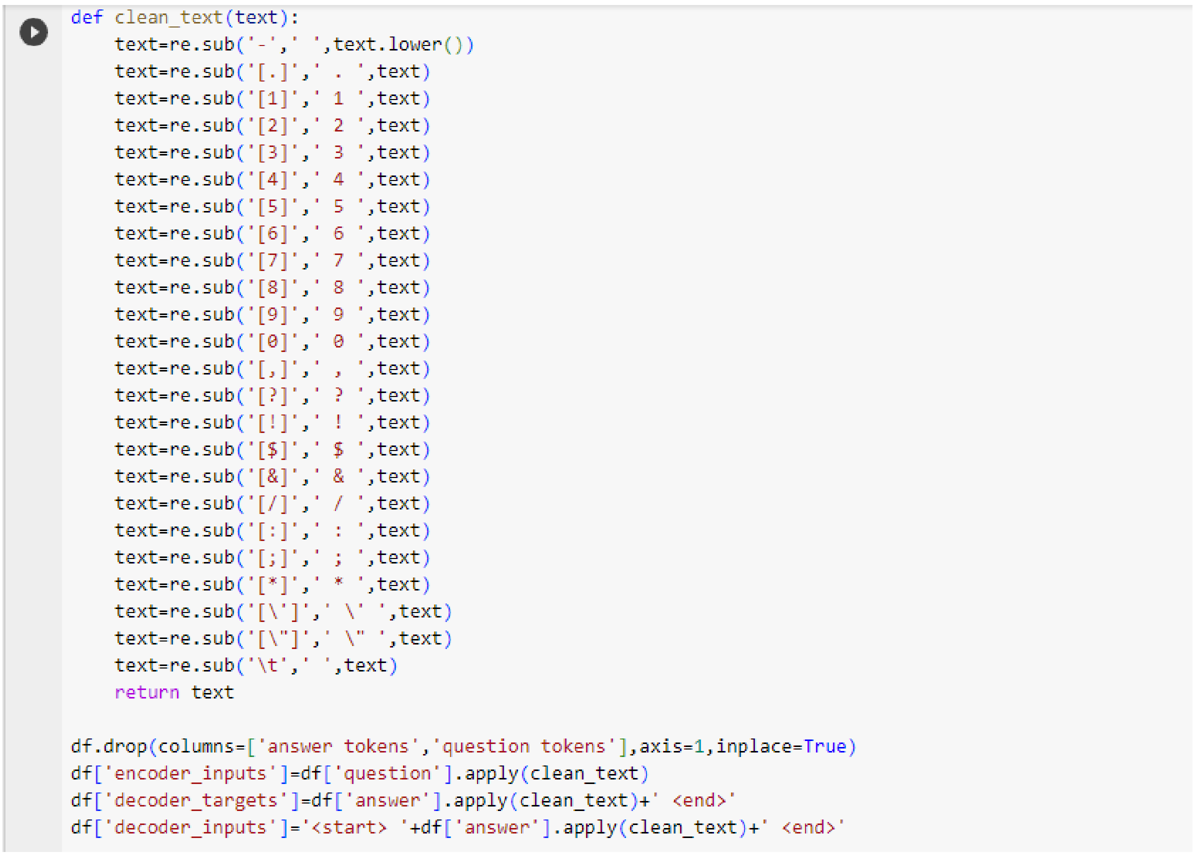
```python
df['question tokens']=df['question'].apply(lambda x:len(x.split()))
df['answer tokens']=df['answer'].apply(lambda x:len(x.split()))
plt.style.use('fivethirtyeight')
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))
sns.set_palette('Set2')
sns.histplot(x=df['question tokens'],data=df,kde=True,ax=ax[0])
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()
```



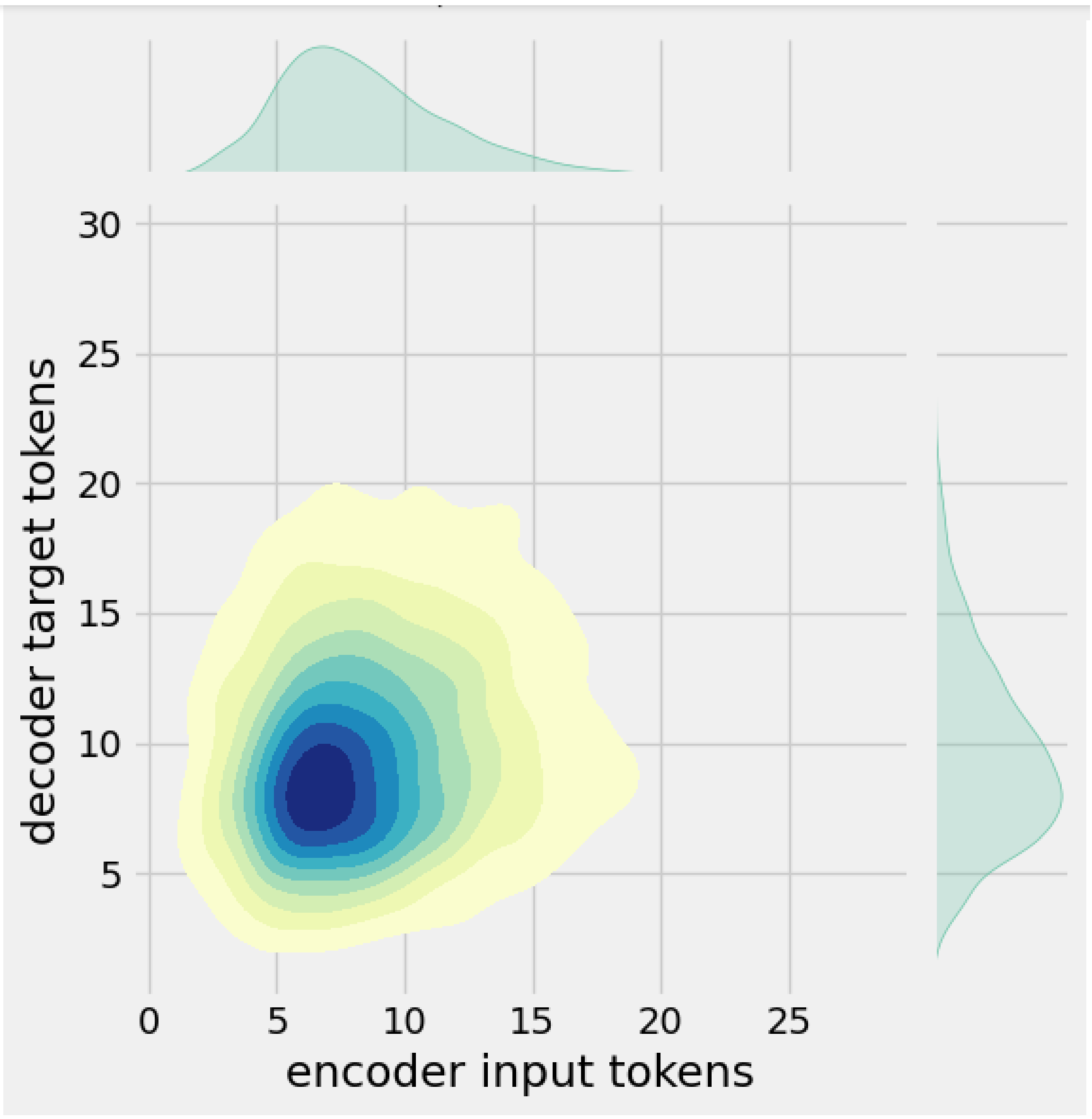


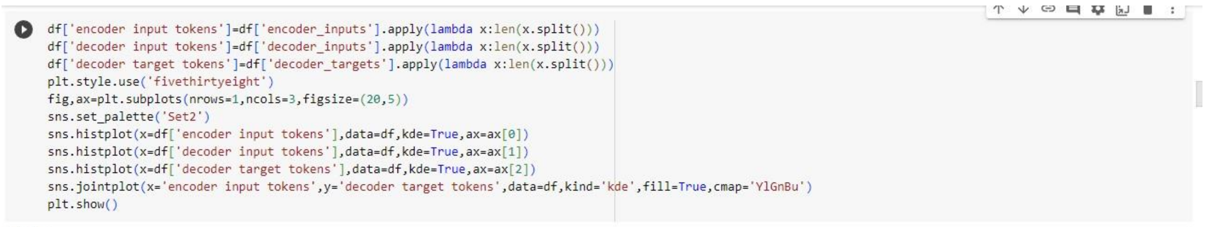
```python
df.head(10)
```

| | 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> |

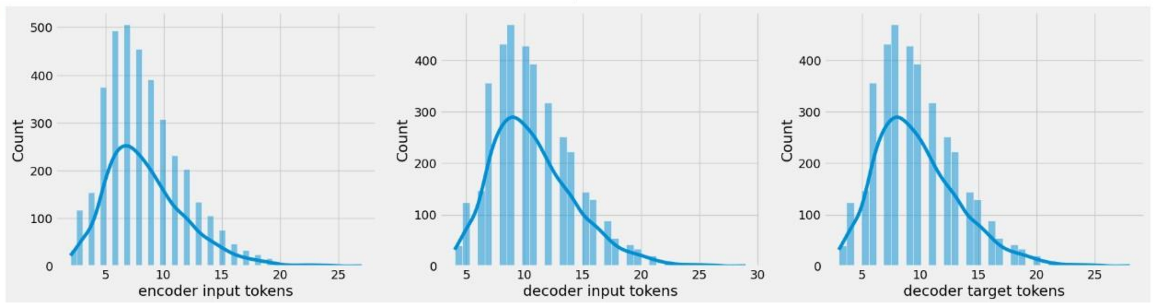


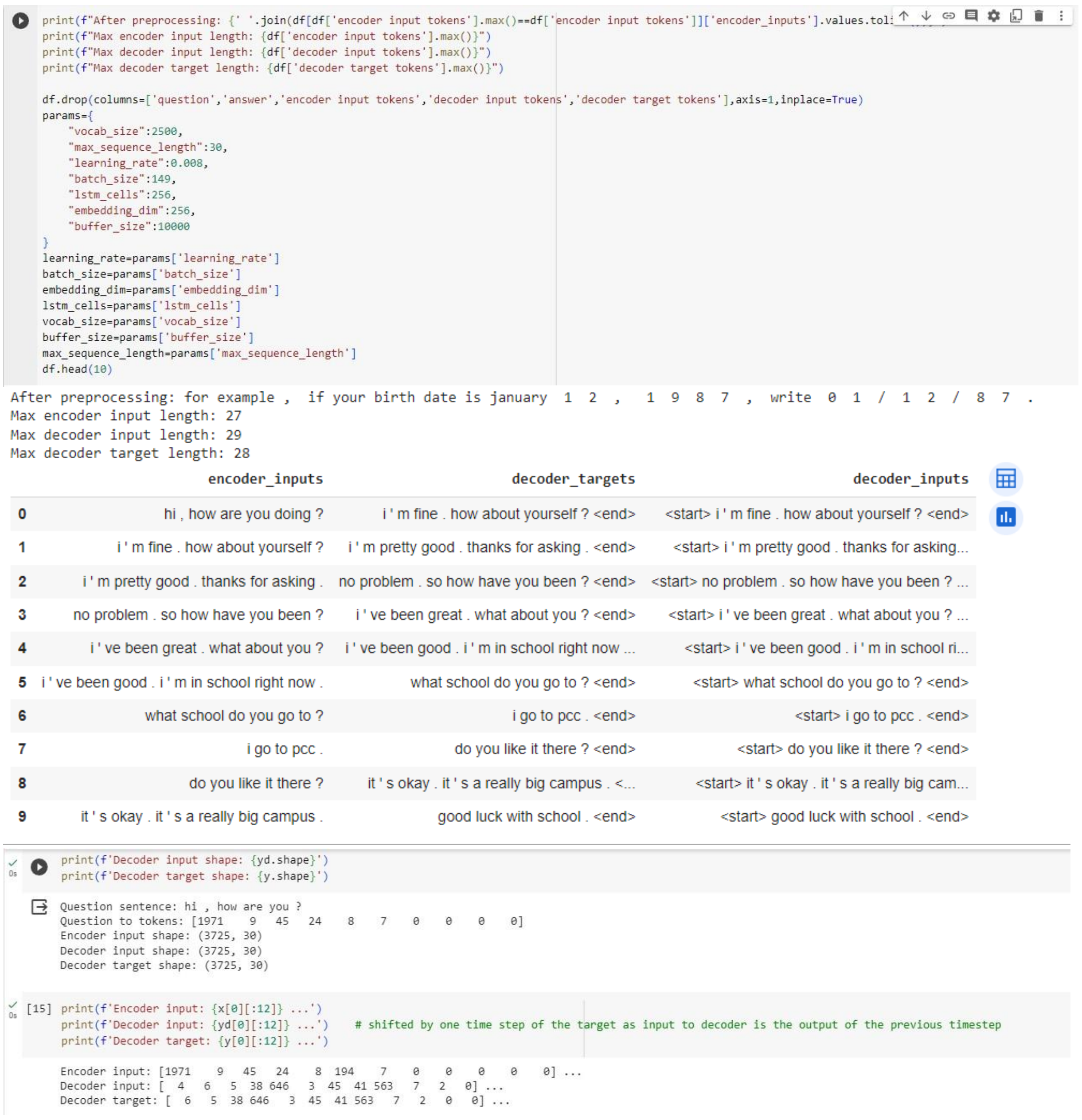
```
def clean_text(text):
    text=re.sub('-',' ',text.lower())
    text=re.sub('[.]',' . ',text)
    text=re.sub('[1]',' 1 ',text)
    text=re.sub('[2]',' 2 ',text)
    text=re.sub('[3]',' 3 ',text)
    text=re.sub('[4]',' 4 ',text)
    text=re.sub('[5]',' 5 ',text)
    text=re.sub('[6]',' 6 ',text)
    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['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.toli
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
```

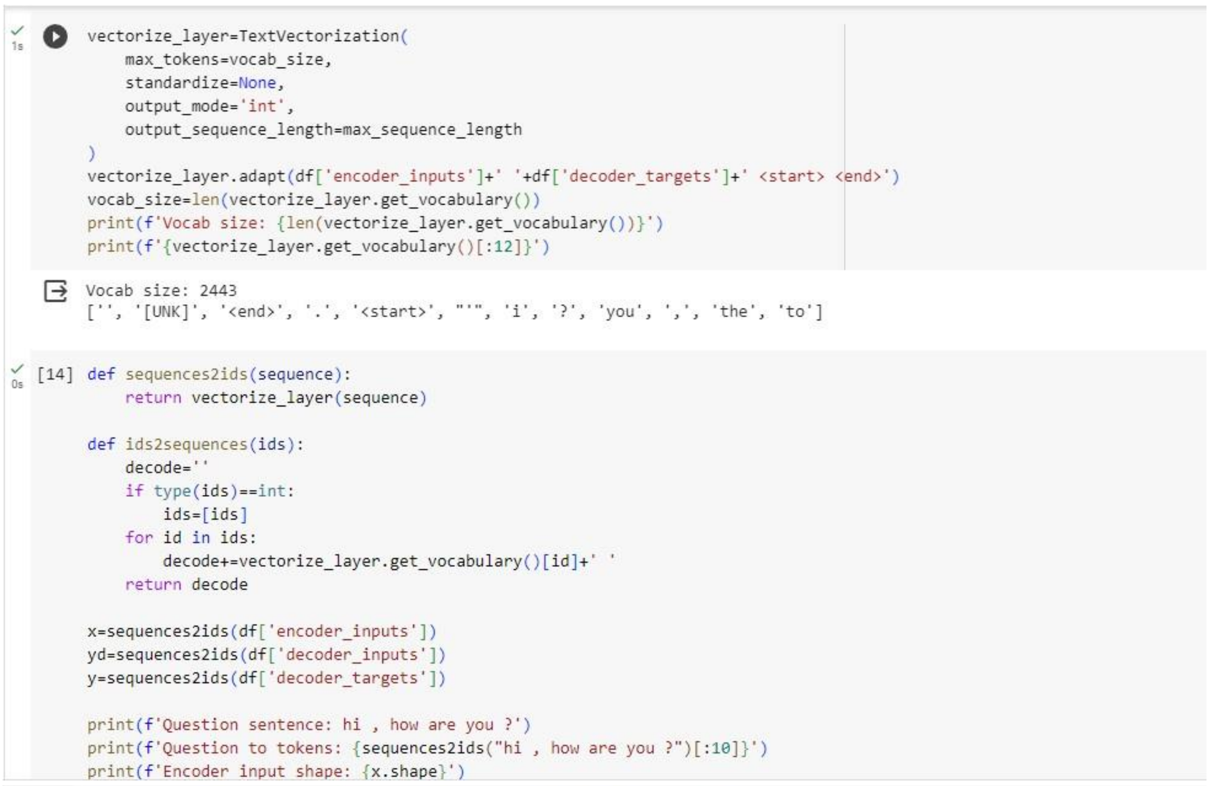
| | encoder_inputs | decoder_targets | decoder_inputs |
|---|---|---|---|
| **0** | hi , how are you doing ? | i ' m fine . how about yourself ? <end> | <start> i ' m fine . how about yourself ? <end> |
| **1** | i ' m fine . how about yourself ? | i ' m pretty good . thanks for asking . <end> | <start> 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> |

```
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)
```

```
[15] 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]} ...')
```

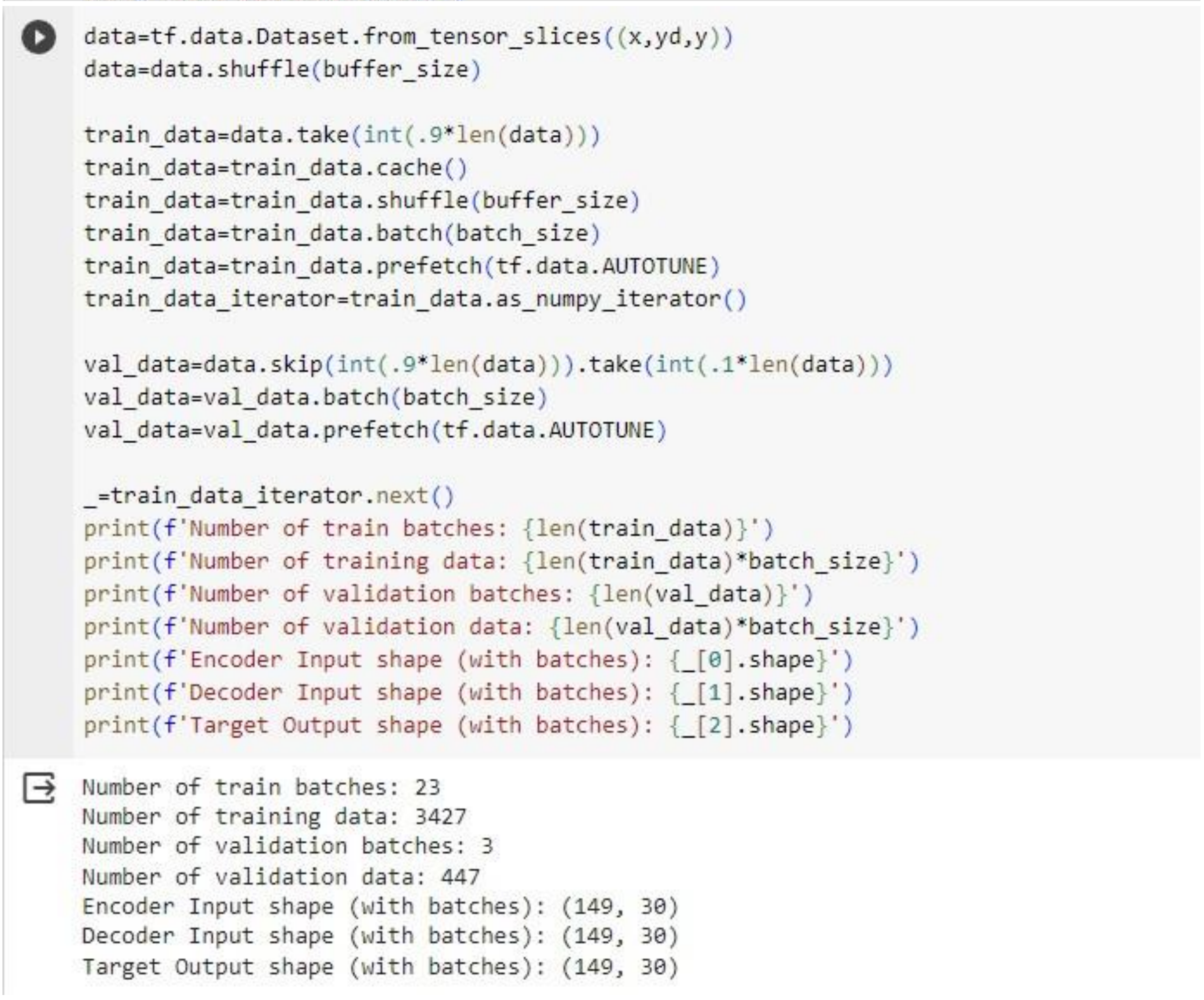
```
Encoder input: [1971    9   45   24    8  194    7    0    0    0    0    0] ...
Decoder input: [  4    6    5  38 646    3  45  41 563    7    2    0] ...
Decoder target: [  6    5  38 646    3  45  41 563    7    2    0    0] ...
```



```
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]}')
```

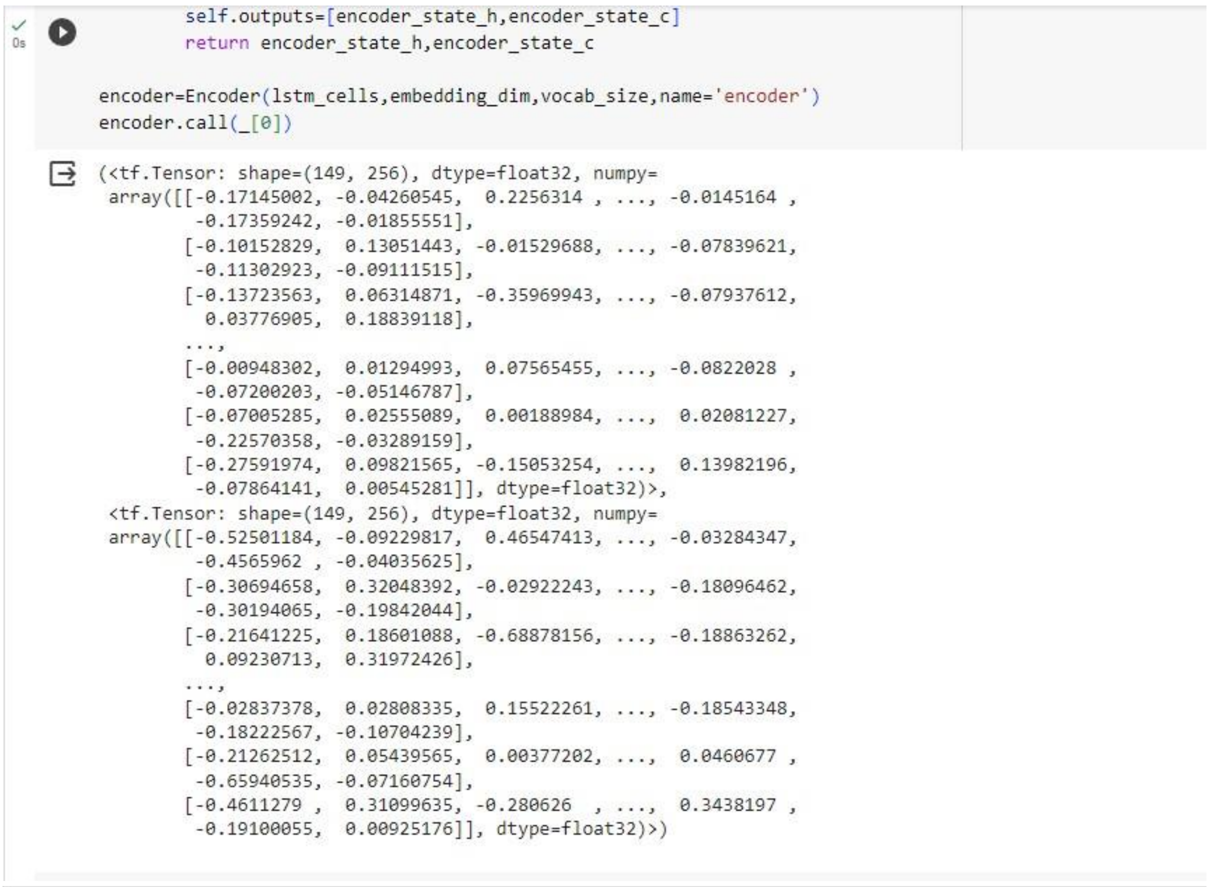
```
Vocab size: 2443
['', '[UNK]', '<end>', '.', '<start>', "'", 'i', '?', 'you', ',', 'the', 'to']
```

```
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}')
```



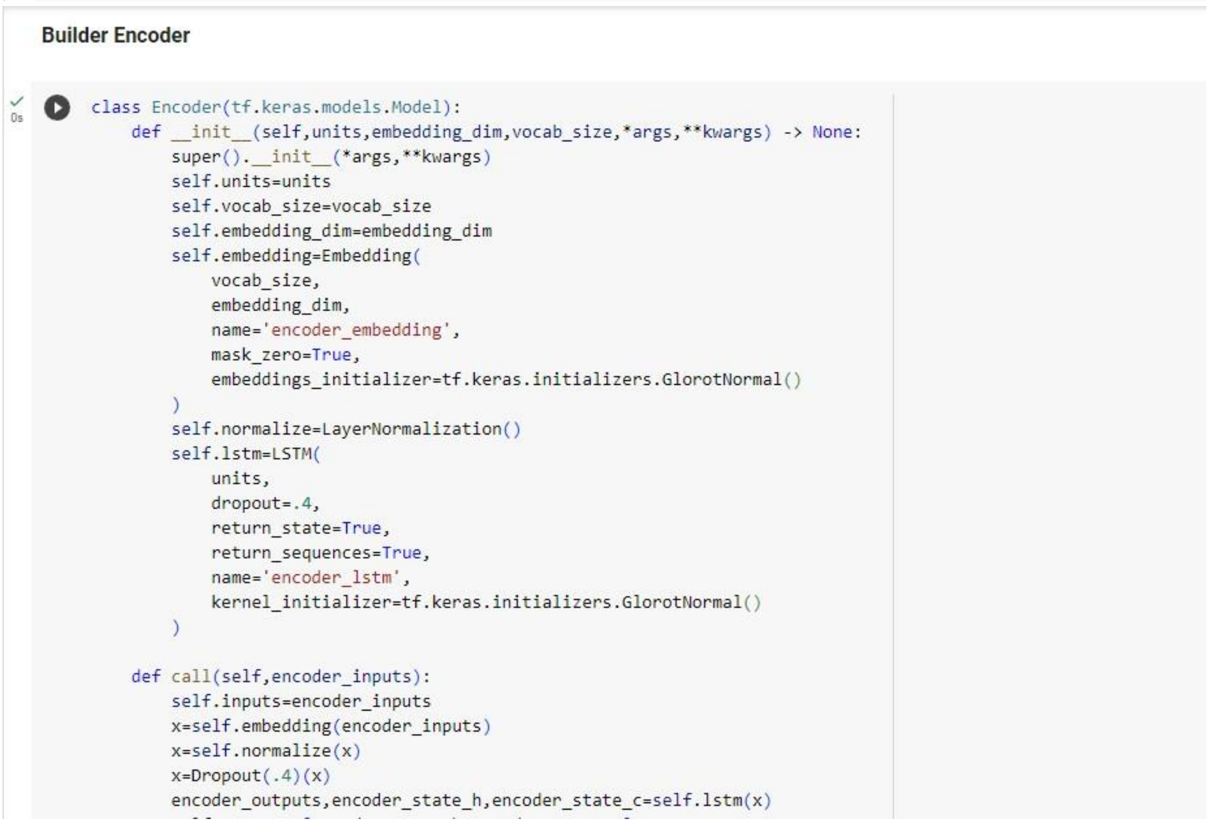
```
data=tf.data.Dataset.from_tensor_slices((x,yd,y))
data=data.shuffle(buffer_size)

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)
```



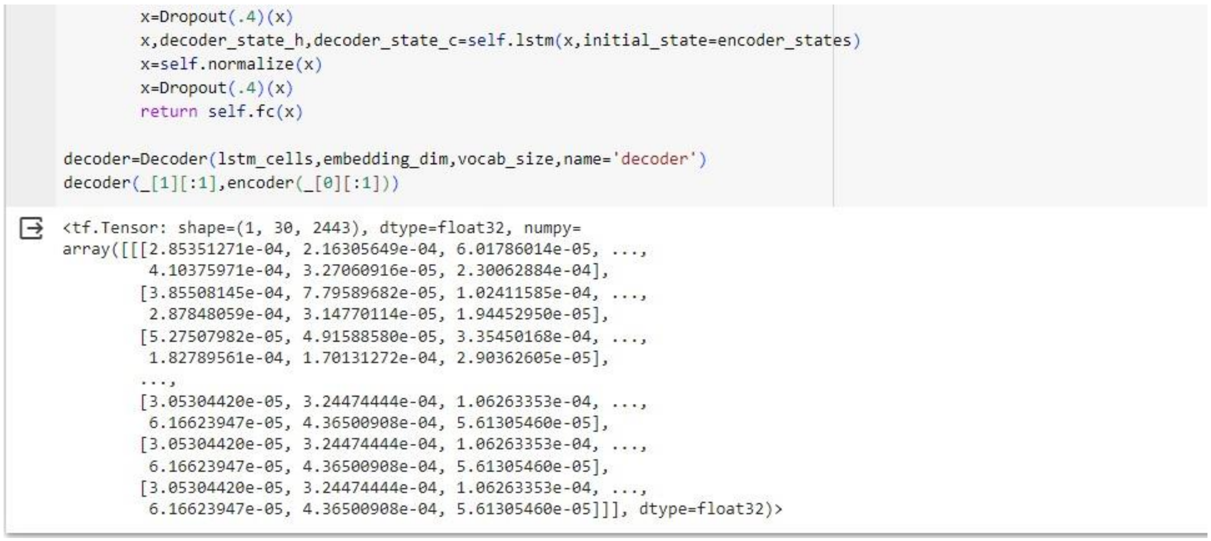
```
        self.outputs=[encoder_state_h,encoder_state_c]
        return encoder_state_h,encoder_state_c

encoder=Encoder(lstm_cells,embedding_dim,vocab_size,name='encoder')
encoder.call(_[0])
```

```
(<tf.Tensor: shape=(149, 256), dtype=float32, numpy=
 array([[-0.17145002, -0.04260545,  0.2256314 , ..., -0.0145164 ,
         -0.17359242, -0.01855551],
        [-0.10152829,  0.13051443, -0.01529688, ..., -0.07839621,
         -0.11302923, -0.09111515],
        [-0.13723563,  0.06314871, -0.35969943, ..., -0.07937612,
          0.03776905,  0.18839118],
        ...,
        [-0.00948302,  0.01294993,  0.07565455, ..., -0.0822028 ,
         -0.07200203, -0.05146787],
        [-0.07005285,  0.02555089,  0.00188984, ...,  0.02081227,
         -0.22570358, -0.03289159],
        [-0.27591974,  0.09821565, -0.15053254, ...,  0.13982196,
         -0.07864141,  0.00545281]], dtype=float32)>,
 <tf.Tensor: shape=(149, 256), dtype=float32, numpy=
 array([[-0.52501184, -0.09229817,  0.46547413, ..., -0.03284347,
         -0.4565962 , -0.04035625],
        [-0.30694658,  0.32048392, -0.02922243, ..., -0.18096462,
         -0.30194065, -0.19842044],
        [-0.21641225,  0.18601088, -0.68878156, ..., -0.18863262,
          0.09230713,  0.31972426],
        ...,
        [-0.02837378,  0.02808335,  0.15522261, ..., -0.18543348,
         -0.18222567, -0.10704239],
        [-0.21262512,  0.05439565,  0.00377202, ...,  0.0460677 ,
         -0.65940535, -0.07160754],
        [-0.4611279 ,  0.31099635, -0.280626  , ...,  0.3438197 ,
         -0.19100055,  0.00925176]], dtype=float32)>)
```



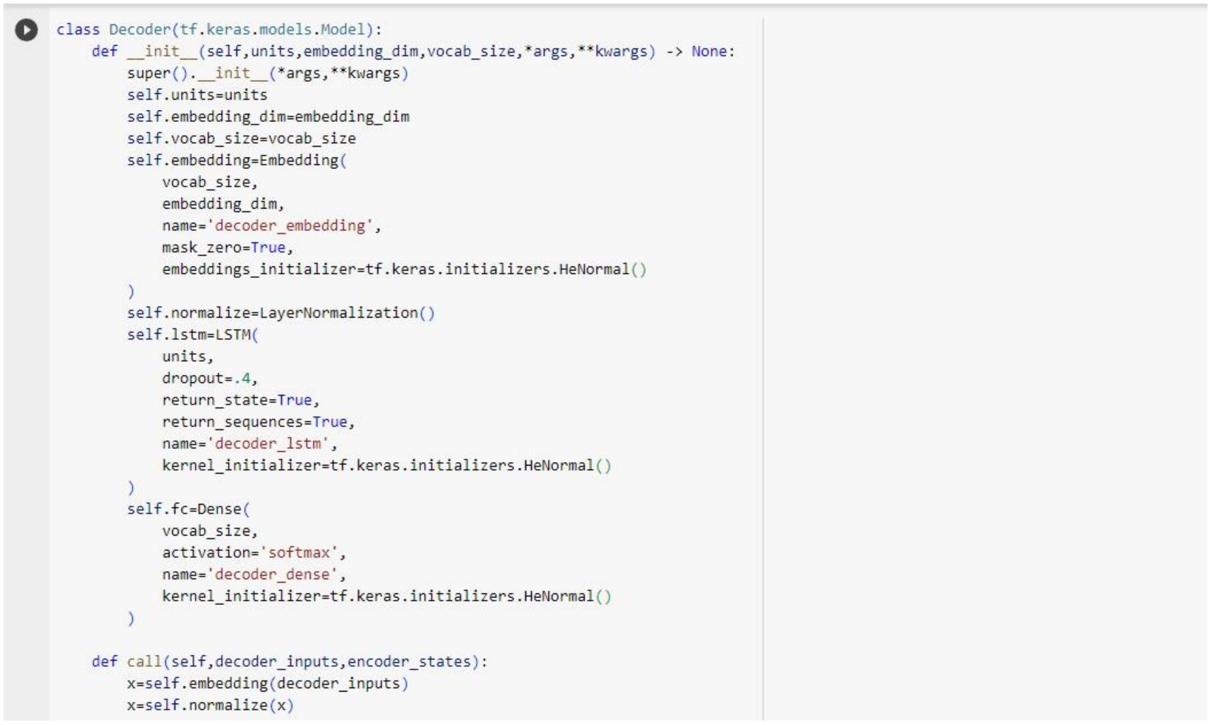
**Builder Encoder**

```
class Encoder(tf.keras.models.Model):
    def __init__(self,units,embedding_dim,vocab_size,*args,**kwargs) -> None:
        super().__init__(*args,**kwargs)
        self.units=units
        self.vocab_size=vocab_size
        self.embedding_dim=embedding_dim
        self.embedding=Embedding(
            vocab_size,
            embedding_dim,
            name='encoder_embedding',
            mask_zero=True,
            embeddings_initializer=tf.keras.initializers.GlorotNormal()
        )
        self.normalize=LayerNormalization()
        self.lstm=LSTM(
            units,
            dropout=.4,
            return_state=True,
            return_sequences=True,
            name='encoder_lstm',
            kernel_initializer=tf.keras.initializers.GlorotNormal()
        )

    def call(self,encoder_inputs):
        self.inputs=encoder_inputs
        x=self.embedding(encoder_inputs)
        x=self.normalize(x)
        x=Dropout(.4)(x)
        encoder_outputs,encoder_state_h,encoder_state_c=self.lstm(x)
```



```
        x=Dropout(.4)(x)
        x,decoder_state_h,decoder_state_c=self.lstm(x,initial_state=encoder_states)
        x=self.normalize(x)
        x=Dropout(.4)(x)
        return self.fc(x)

decoder=Decoder(lstm_cells,embedding_dim,vocab_size,name='decoder')
decoder(_[1][:1],encoder(_[0][:1]))
```

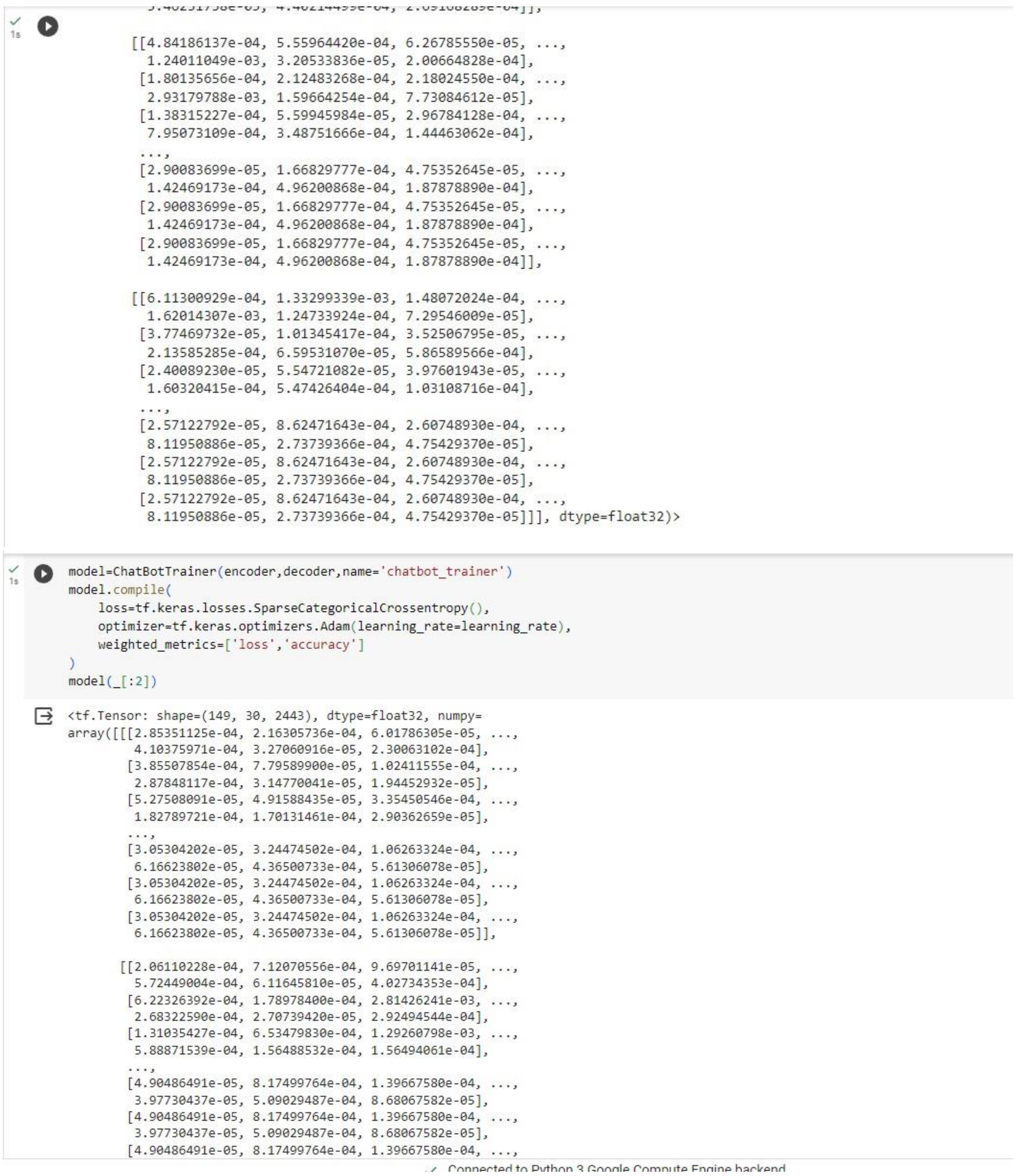
```
<tf.Tensor: shape=(1, 30, 2443), dtype=float32, numpy=
array([[[2.85351271e-04, 2.16305649e-04, 6.01786014e-05, ...,
         4.10375971e-04, 3.27060916e-05, 2.30062884e-04],
        [3.85508145e-04, 7.79589682e-05, 1.02411585e-04, ...,
         2.87848059e-04, 3.14770114e-05, 1.94452950e-05],
        [5.27507982e-05, 4.91588580e-05, 3.35450168e-04, ...,
         1.82789561e-04, 1.70131272e-04, 2.90362605e-05],
        ...,
        [3.05304420e-05, 3.24474444e-04, 1.06263353e-04, ...,
         6.16623947e-05, 4.36500908e-04, 5.61305460e-05],
        [3.05304420e-05, 3.24474444e-04, 1.06263353e-04, ...,
         6.16623947e-05, 4.36500908e-04, 5.61305460e-05],
        [3.05304420e-05, 3.24474444e-04, 1.06263353e-04, ...,
         6.16623947e-05, 4.36500908e-04, 5.61305460e-05]]], dtype=float32)>
```



```
class Decoder(tf.keras.models.Model):
    def __init__(self,units,embedding_dim,vocab_size,*args,**kwargs) -> None:
        super().__init__(*args,**kwargs)
        self.units=units
        self.embedding_dim=embedding_dim
        self.vocab_size=vocab_size
        self.embedding=Embedding(
            vocab_size,
            embedding_dim,
            name='decoder_embedding',
            mask_zero=True,
            embeddings_initializer=tf.keras.initializers.HeNormal()
        )
        self.normalize=LayerNormalization()
        self.lstm=LSTM(
            units,
            dropout=.4,
            return_state=True,
            return_sequences=True,
            name='decoder_lstm',
            kernel_initializer=tf.keras.initializers.HeNormal()
        )
        self.fc=Dense(
            vocab_size,
            activation='softmax',
            name='decoder_dense',
            kernel_initializer=tf.keras.initializers.HeNormal()
        )

    def call(self,decoder_inputs,encoder_states):
        x=self.embedding(decoder_inputs)
        x=self.normalize(x)
    def train_step(self,batch):
        encoder_inputs,decoder_inputs,y=batch
        with tf.GradientTape() as tape:
            encoder_states=self.encoder(encoder_inputs,training=True)
            y_pred=self.decoder(decoder_inputs,encoder_states,training=True)
            loss=self.loss_fn(y,y_pred)
            acc=self.accuracy_fn(y,y_pred)

        variables=self.encoder.trainable_variables+self.decoder.trainable_variables
        grads=tape.gradient(loss,variables)
        self.optimizer.apply_gradients(zip(grads,variables))
        metrics={'loss':loss,'accuracy':acc}
        return metrics

    def test_step(self,batch):
        encoder_inputs,decoder_inputs,y=batch
        encoder_states=self.encoder(encoder_inputs,training=True)
        y_pred=self.decoder(decoder_inputs,encoder_states,training=True)
        loss=self.loss_fn(y,y_pred)
        acc=self.accuracy_fn(y,y_pred)
        metrics={'loss':loss,'accuracy':acc}
        return metrics
```

**Build Training Model**

```
class ChatBotTrainer(tf.keras.models.Model):
    def __init__(self,encoder,decoder,*args,**kwargs):
        super().__init__(*args,**kwargs)
        self.encoder=encoder
        self.decoder=decoder

    def loss_fn(self,y_true,y_pred):
        loss=self.loss(y_true,y_pred)
        mask=tf.math.logical_not(tf.math.equal(y_true,0))
        mask=tf.cast(mask,dtype=loss.dtype)
        loss*=mask
        return tf.reduce_mean(loss)

    def accuracy_fn(self,y_true,y_pred):
        pred_values = tf.cast(tf.argmax(y_pred, axis=-1), dtype='int64')
        correct = tf.cast(tf.equal(y_true, pred_values), dtype='float64')
        mask = tf.cast(tf.greater(y_true, 0), dtype='float64')
        n_correct = tf.keras.backend.sum(mask * correct)
        n_total = tf.keras.backend.sum(mask)
        return n_correct / n_total

    def call(self,inputs):
        encoder_inputs,decoder_inputs=inputs
        encoder_states=self.encoder(encoder_inputs)
        return self.decoder(decoder_inputs,encoder_states)
```



```
        5.40231758e-05, 4.40214499e-04, 2.09108289e-04]],

       [[4.84186137e-04, 5.55964420e-04, 6.26785550e-05, ...,
         1.24011049e-03, 3.20533836e-05, 2.00664828e-04],
        [1.80135656e-04, 2.12483268e-04, 2.18024550e-04, ...,
         2.93179788e-03, 1.59664254e-04, 7.73084612e-05],
        [1.38315227e-04, 5.59945984e-05, 2.96784128e-04, ...,
         7.95073109e-04, 3.48751666e-04, 1.44463062e-04],
        ...,
        [2.90083699e-05, 1.66829777e-04, 4.75352645e-05, ...,
         1.42469173e-04, 4.96200868e-04, 1.87878890e-04],
        [2.90083699e-05, 1.66829777e-04, 4.75352645e-05, ...,
         1.42469173e-04, 4.96200868e-04, 1.87878890e-04],
        [2.90083699e-05, 1.66829777e-04, 4.75352645e-05, ...,
         1.42469173e-04, 4.96200868e-04, 1.87878890e-04]],

       [[6.11300929e-04, 1.33299339e-03, 1.48072024e-04, ...,
         1.62014307e-03, 1.24733924e-04, 7.29546009e-05],
        [3.77469732e-05, 1.01345417e-04, 3.52506795e-05, ...,
         2.13585285e-04, 6.59531070e-05, 5.86589566e-04],
        [2.40089230e-05, 5.54721082e-05, 3.97601943e-05, ...,
         1.60320415e-04, 5.47426404e-04, 1.03108716e-04],
        ...,
        [2.57122792e-05, 8.62471643e-04, 2.60748930e-04, ...,
         8.11950886e-05, 2.73739366e-04, 4.75429370e-05],
        [2.57122792e-05, 8.62471643e-04, 2.60748930e-04, ...,
         8.11950886e-05, 2.73739366e-04, 4.75429370e-05],
        [2.57122792e-05, 8.62471643e-04, 2.60748930e-04, ...,
         8.11950886e-05, 2.73739366e-04, 4.75429370e-05]]], dtype=float32)>
```

```python
model=ChatBotTrainer(encoder,decoder,name='chatbot_trainer')
model.compile(
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
    weighted_metrics=['loss','accuracy']
)
model(_[:2])
```

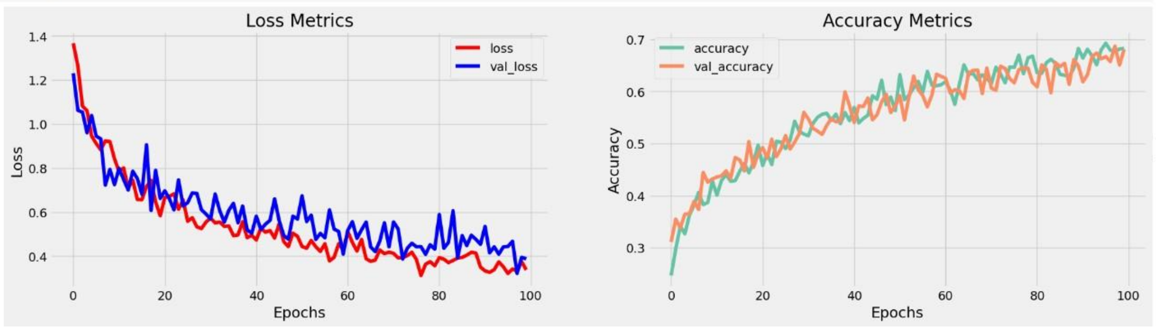
```
<tf.Tensor: shape=(149, 30, 2443), dtype=float32, numpy=
array([[[2.85351125e-04, 2.16305736e-04, 6.01786305e-05, ...,
         4.10375971e-04, 3.27060916e-05, 2.30063102e-04],
        [3.85507854e-04, 7.79589900e-05, 1.02411555e-04, ...,
         2.87848117e-04, 3.14770041e-05, 1.94452932e-05],
        [5.27508091e-05, 4.91588435e-05, 3.35450546e-04, ...,
         1.82789721e-04, 1.70131461e-04, 2.90362659e-05],
        ...,
        [3.05304202e-05, 3.24474502e-04, 1.06263324e-04, ...,
         6.16623802e-05, 4.36500733e-04, 5.61306078e-05],
        [3.05304202e-05, 3.24474502e-04, 1.06263324e-04, ...,
         6.16623802e-05, 4.36500733e-04, 5.61306078e-05],
        [3.05304202e-05, 3.24474502e-04, 1.06263324e-04, ...,
         6.16623802e-05, 4.36500733e-04, 5.61306078e-05]],

       [[2.06110228e-04, 7.12070556e-04, 9.69701141e-05, ...,
         5.72449004e-04, 6.11645810e-05, 4.02734353e-04],
        [6.22326392e-04, 1.78978400e-04, 2.81426241e-03, ...,
         2.68322590e-04, 2.70739420e-05, 2.92494544e-04],
        [1.31035427e-04, 6.53479830e-04, 1.29260798e-03, ...,
         5.88871539e-04, 1.56488532e-04, 1.56494061e-04],
        ...,
        [4.90486491e-05, 8.17499764e-04, 1.39667580e-04, ...,
         3.97730437e-05, 5.09029487e-04, 8.68067582e-05],
        [4.90486491e-05, 8.17499764e-04, 1.39667580e-04, ...,
         3.97730437e-05, 5.09029487e-04, 8.68067582e-05],
        [4.90486491e-05, 8.17499764e-04, 1.39667580e-04, ...,
```

## Train Model

```
history=model.fit(
    train_data,
    epochs=100,
    validation_data=val_data,
    callbacks=[
        tf.keras.callbacks.TensorBoard(log_dir='logs'),
        tf.keras.callbacks.ModelCheckpoint('ckpt',verbose=1,save_best_only=True)
    ]
)
```

```
23/23 [==============================] - 38s 2s/step - loss: 0.3605 - accuracy: 0.6601 - val_loss: 0.4476 - val_accuracy: 0.6538
Epoch 88/100
23/23 [==============================] - ETA: 0s - loss: 0.3585 - accuracy: 0.6644
Epoch 88: val_loss did not improve from 0.38672
23/23 [==============================] - 38s 2s/step - loss: 0.3609 - accuracy: 0.6629 - val_loss: 0.4946 - val_accuracy: 0.6134
Epoch 89/100
23/23 [==============================] - ETA: 0s - loss: 0.3559 - accuracy: 0.6634
Epoch 89: val_loss did not improve from 0.38672
23/23 [==============================] - 38s 2s/step - loss: 0.3583 - accuracy: 0.6626 - val_loss: 0.4761 - val_accuracy: 0.6609
Epoch 90/100
23/23 [==============================] - ETA: 0s - loss: 0.3540 - accuracy: 0.6649
Epoch 90: val_loss did not improve from 0.38672
23/23 [==============================] - 39s 2s/step - loss: 0.3539 - accuracy: 0.6656 - val_loss: 0.4536 - val_accuracy: 0.6490
Epoch 91/100
23/23 [==============================] - ETA: 0s - loss: 0.3512 - accuracy: 0.6679
Epoch 91: val_loss did not improve from 0.38672
23/23 [==============================] - 38s 2s/step - loss: 0.3504 - accuracy: 0.6677 - val_loss: 0.5347 - val_accuracy: 0.6182
Epoch 92/100
23/23 [==============================] - ETA: 0s - loss: 0.3526 - accuracy: 0.6681
Epoch 92: val_loss did not improve from 0.38672
23/23 [==============================] - 37s 2s/step - loss: 0.3515 - accuracy: 0.6686 - val_loss: 0.4149 - val_accuracy: 0.6321
Epoch 93/100
```

```
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))
ax[0].plot(history.history['loss'],label='loss',c='red')
ax[0].plot(history.history['val_loss'],label='val_loss',c = 'blue')
ax[0].set_xlabel('Epochs')
ax[1].set_xlabel('Epochs')
ax[0].set_ylabel('Loss')
ax[1].set_ylabel('Accuracy')
ax[0].set_title('Loss Metrics')
ax[1].set_title('Accuracy Metrics')
ax[1].plot(history.history['accuracy'],label='accuracy')
ax[1].plot(history.history['val_accuracy'],label='val_accuracy')
ax[0].legend()
ax[1].legend()
plt.show()
```



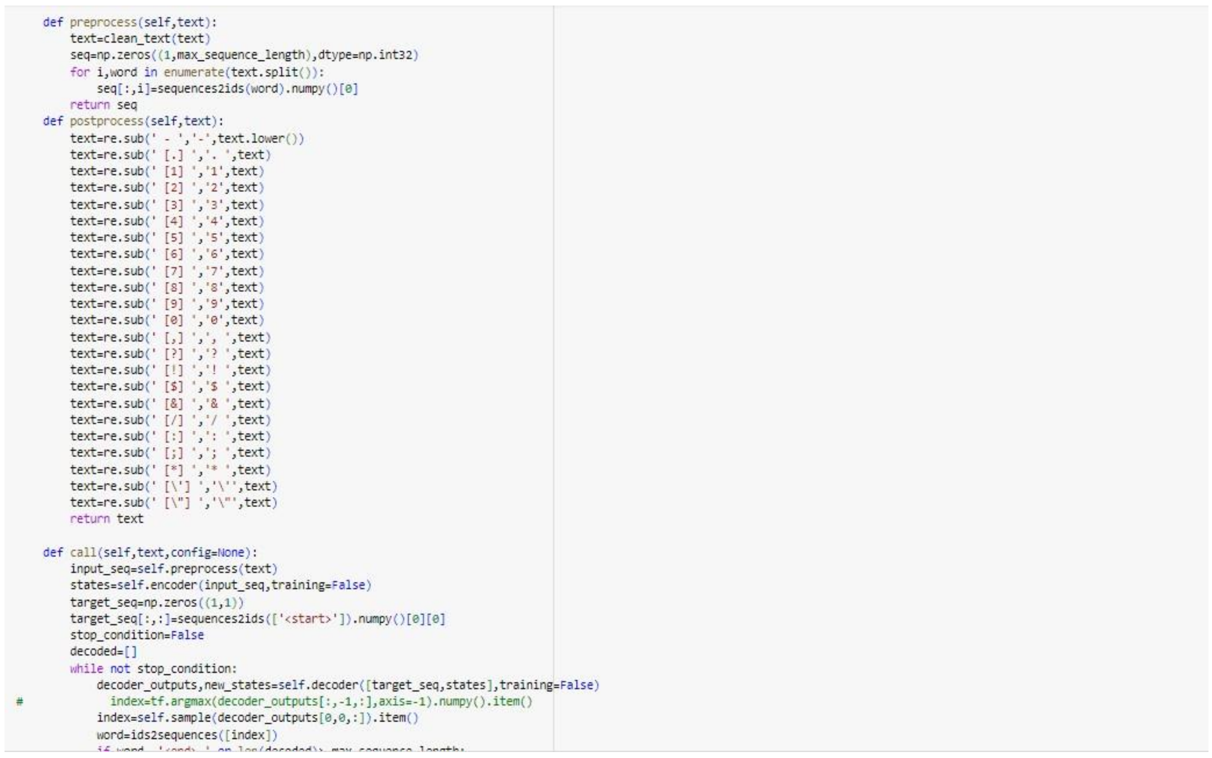
### Save Model

```
model.load_weights('ckpt')
model.save('models',save_format='tf')
```

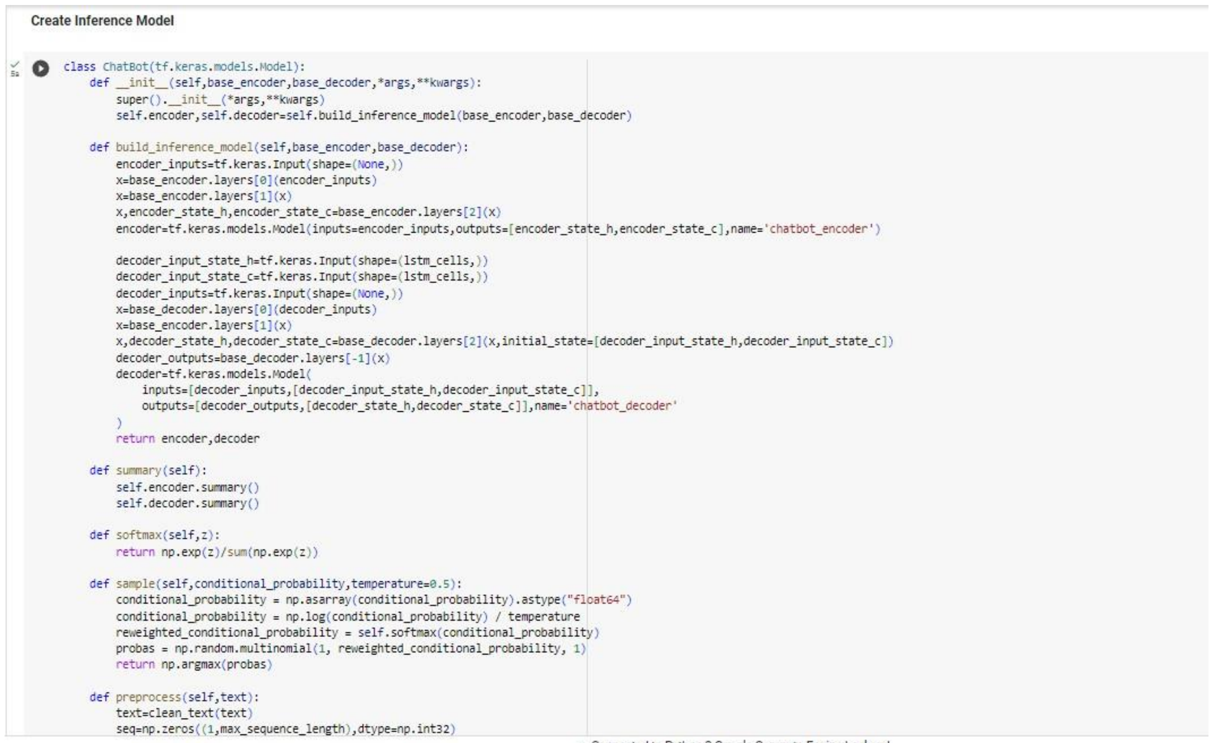
```
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config.
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config.
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config.
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config.
```

```
[25] for idx,i in enumerate(model.layers):
         print('Encoder layers:' if idx==0 else 'Decoder layers: ')
         for j in i.layers:
             print(j)
         print('--------------------')
```

```
Encoder layers:
<keras.src.layers.core.embedding.Embedding object at 0x792cb59dece0>
<keras.src.layers.normalization.layer_normalization.LayerNormalization object at 0x792cb59df880>
<keras.src.layers.rnn.lstm.LSTM object at 0x792cb376eb00>
--------------------
Decoder layers:
<keras.src.layers.core.embedding.Embedding object at 0x792cb35e3910>
<keras.src.layers.normalization.layer_normalization.LayerNormalization object at 0x792cb35e1900>
<keras.src.layers.rnn.lstm.LSTM object at 0x792cb367d810>
<keras.src.layers.core.dense.Dense object at 0x792cb35ac460>
--------------------
```

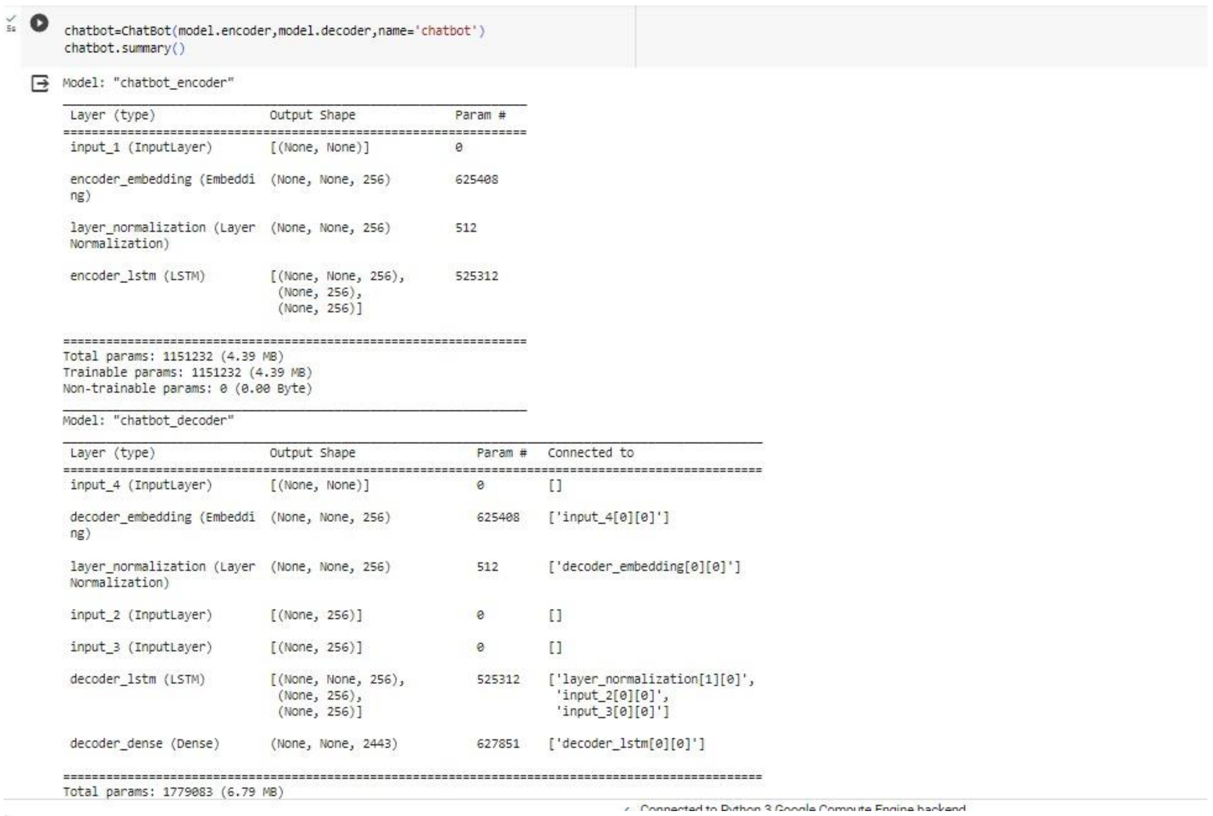


```
    def preprocess(self,text):
        text=clean_text(text)
        seq=np.zeros((1,max_sequence_length),dtype=np.int32)
        for i,word in enumerate(text.split()):
            seq[:,i]=sequences2ids(word).numpy()[0]
        return seq
    def postprocess(self,text):
        text=re.sub(' - ','-',text.lower())
        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)
        return text

    def call(self,text,config=None):
        input_seq=self.preprocess(text)
        states=self.encoder(input_seq,training=False)
        target_seq=np.zeros((1,1))
        target_seq[:,:]=sequences2ids(['<start>']).numpy()[0][0]
        stop_condition=False
        decoded=[]
        while not stop_condition:
            decoder_outputs,new_states=self.decoder([target_seq,states],training=False)
#             index=tf.argmax(decoder_outputs[:,-1,:],axis=-1).numpy().item()
            index=self.sample(decoder_outputs[0,0,:]).item()
            word=ids2sequences([index])
```



**Create Inference Model**

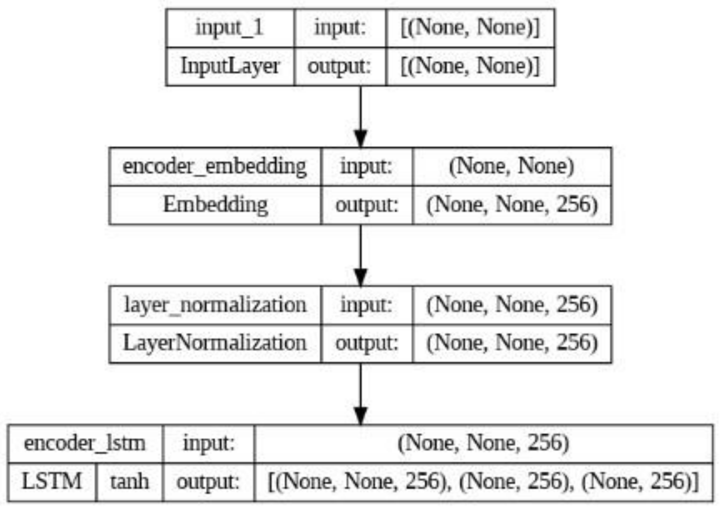
```
class ChatBot(tf.keras.models.Model):
    def __init__(self,base_encoder,base_decoder,*args,**kwargs):
        super().__init__(*args,**kwargs)
        self.encoder,self.decoder=self.build_inference_model(base_encoder,base_decoder)

    def build_inference_model(self,base_encoder,base_decoder):
        encoder_inputs=tf.keras.Input(shape=(None,))
        x=base_encoder.layers[0](encoder_inputs)
        x=base_encoder.layers[1](x)
        x,encoder_state_h,encoder_state_c=base_encoder.layers[2](x)
        encoder=tf.keras.models.Model(inputs=encoder_inputs,outputs=[encoder_state_h,encoder_state_c],name='chatbot_encoder')

        decoder_input_state_h=tf.keras.Input(shape=(lstm_cells,))
        decoder_input_state_c=tf.keras.Input(shape=(lstm_cells,))
        decoder_inputs=tf.keras.Input(shape=(None,))
        x=base_decoder.layers[0](decoder_inputs)
        x=base_encoder.layers[1](x)
        x,decoder_state_h,decoder_state_c=base_decoder.layers[2](x,initial_state=[decoder_input_state_h,decoder_input_state_c])
        decoder_outputs=base_decoder.layers[-1](x)
        decoder=tf.keras.models.Model(
            inputs=[decoder_inputs,[decoder_input_state_h,decoder_input_state_c]],
            outputs=[decoder_outputs,[decoder_state_h,decoder_state_c]],name='chatbot_decoder'
        )
        return encoder,decoder

    def summary(self):
        self.encoder.summary()
        self.decoder.summary()

    def softmax(self,z):
        return np.exp(z)/sum(np.exp(z))

    def sample(self,conditional_probability,temperature=0.5):
        conditional_probability = np.asarray(conditional_probability).astype("float64")
        conditional_probability = np.log(conditional_probability) / temperature
        reweighted_conditional_probability = self.softmax(conditional_probability)
        probas = np.random.multinomial(1, reweighted_conditional_probability, 1)
        return np.argmax(probas)

    def preprocess(self,text):
        text=clean_text(text)
        seq=np.zeros((1,max_sequence_length),dtype=np.int32)
```



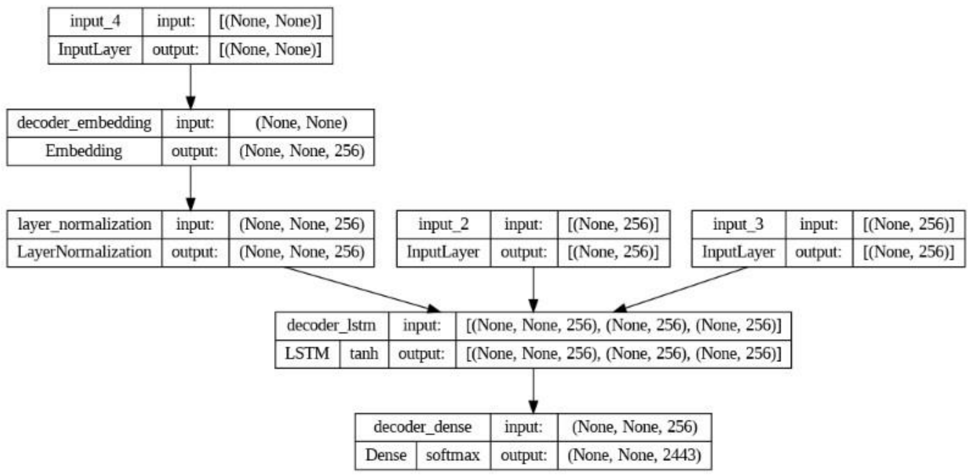
```
chatbot=ChatBot(model.encoder,model.decoder,name='chatbot')
chatbot.summary()
```

```
Model: "chatbot_encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None)]            0

 encoder_embedding (Embeddi  (None, None, 256)         625408
 ng)

 layer_normalization (Layer  (None, None, 256)         512
 Normalization)

 encoder_lstm (LSTM)         [(None, None, 256),       525312
                              (None, 256),
                              (None, 256)]

=================================================================
Total params: 1151232 (4.39 MB)
Trainable params: 1151232 (4.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Model: "chatbot_decoder"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_4 (InputLayer)        [(None, None)]               0         []

 decoder_embedding (Embeddi  (None, None, 256)            625408    ['input_4[0][0]']
 ng)

 layer_normalization (Layer  (None, None, 256)            512       ['decoder_embedding[0][0]']
 Normalization)

 input_2 (InputLayer)        [(None, 256)]                0         []

 input_3 (InputLayer)        [(None, 256)]                0         []

 decoder_lstm (LSTM)         [(None, None, 256),          525312    ['layer_normalization[1][0]',
                              (None, 256),                           'input_2[0][0]',
                              (None, 256)]                           'input_3[0][0]']

 decoder_dense (Dense)       (None, None, 2443)           627851    ['decoder_lstm[0][0]']

==================================================================================================
Total params: 1779083 (6.79 MB)
```

```
tf.keras.utils.plot_model(chatbot.encoder,to_file='encoder.png',show_shapes=True,show_layer_activations=True)
```


input_1 | input: | [(None, None)]
InputLayer | output: | [(None, None)]
encoder_embedding | input: | (None, None)
Embedding | output: | (None, None, 256)
layer_normalization | input: | (None, None, 256)
LayerNormalization | output: | (None, None, 256)
encoder_lstm | input: | (None, None, 256)
LSTM | tanh | output: | [(None, None, 256), (None, 256), (None, 256)]

```
tf.keras.utils.plot_model(chatbot.decoder,to_file='decoder.png',show_shapes=True,show_layer_activations=True)
```



input_4 | InputLayer | input: [(None, None)] | output: [(None, None)]
decoder_embedding | Embedding | input: (None, None) | output: (None, None, 256)
layer_normalization | LayerNormalization | input: (None, None, 256) | output: (None, None, 256)
input_2 | InputLayer | input: [(None, 256)] | output: [(None, 256)]
input_3 | InputLayer | input: [(None, 256)] | output: [(None, 256)]
decoder_lstm | LSTM | tanh | input: [(None, None, 256), (None, 256), (None, 256)] | output: [(None, None, 256), (None, 256), (None, 256)]
decoder_dense | Dense | softmax | input: (None, None, 256) | output: (None, None, 2443)

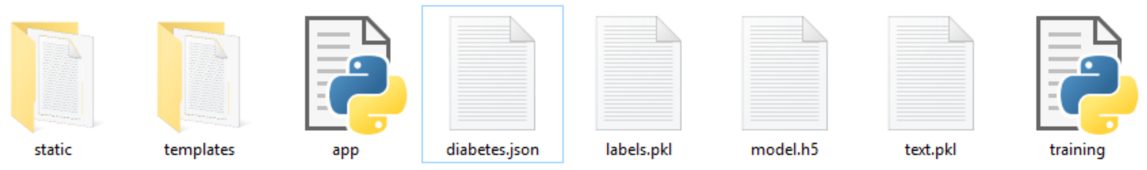
```
[32] def print_conversation(texts):
         for text in texts:
             print(f'You: {text}')
             print(f'Bot: {chatbot(text)}')
             print('========================')
```

```
print_conversation([
    'hi',
    'do yo know me?',
    'what is your name?',
    'you are bot?',
    'hi, how are you doing?',
    "i'm pretty good. thanks for asking.",
    "Don't ever be in a hurry",
    '''I'm gonna put some dirt in your eye ''',
    '''You're trash ''',
    '''I've read all your research on nano-technology ''',
    '''You want forgiveness? Get religion''',
    '''While you're using the bathroom, i'll order some food.''',
    '''Wow! that's terrible.''',
    '''We'll be here forever.''',
    '''I need something that's reliable.''',
    '''A speeding car ran a red light, killing the girl.''',
    '''Tomorrow we'll have rice and fish for lunch.''',
    '''I like this restaurant because they give you free bread.'''
])
```

```
You: hi
Bot: what's the matter?
========================
You: do yo know me?
Bot: the forecast says that it will be warm the weekend.
========================
You: what is your name?
Bot: it's about $10.
========================
You: you are bot?
Bot: i'm not sure.
========================
You: hi, how are you doing?
Bot: i'm going to be a teacher.
========================
You: i'm pretty good. thanks for asking.
Bot: no problem. that's a good idea.
========================
You: Don't ever be in a hurry
Bot: it's the best job.
========================
You: I'm gonna put some dirt in your eye
Bot: that's a good idea.
========================
You: You're trash
Bot: no. it's not too.
```

✓ Connected to Python 3 Google Compute Engine backend

**The Dataset**
The dataset we will be using is ‘diabetes.json’. This is a JSON file that contains the patterns we need to find and the responses we want to return to the user.



Now we are going to build the chatbot using Flask framework but first, let us see the file structure and the type of files we will be creating:

- **diabetes.json** – The data file which has predefined patterns and responses.
- **trainning.py** – In this Python file, we wrote a script to build the model and train our chatbot.
- **Texts.pkl** – This is a pickle file in which we store the words Python object using Nltk that contains a list of our vocabulary.
- **Labels.pkl** – The classes pickle file contains the list of categories(Labels).
- model.h5 – This is the trained model that contains information about the model and has weights of the neurons.
- app.py – This is the flask Python script in which we implemented web-based GUI for our chatbot. Users can easily interact with the bot.

## 1.Import and load the data file

First, make a file name as trainning.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project.

The data file is in JSON format so we used the json package to parse the JSON file into Python. This is how our data.json file looks like.

{"intents": [{"tag": "greeting", "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day", "Whats up"], "responses": ["Hello!", "Good to see you again!", "Hi there, how can I help?"], "context_set": ""}, {"tag": "goodbye", "patterns": ["cya", "See you later", "Goodbye", "I am Leaving", "Have a Good day"], "responses": ["Sad to see you go :(", "Talk to you later", "Goodbye!"], "context_set": ""}, {"tag": "age", "patterns": ["how old", "how old is tim", "what is your age", "how old are you", "age?"], "responses": ["I am 27 years old!", "27 years young!"], "context_set": ""}, {"tag": "name", "patterns": ["what is your name", "what should I call you", "whats your name?"], "responses": ["You can call me Pointer.", "I'm pointer!"], "context_set": ""}, {"tag": "shop", "patterns": ["Id like to buy something", "whats on the menu", "what do you reccommend?", "could i get something to eat"], "responses": ["We sell chocolate chip cookies for $2!", "Cookies are on the menu!"], "context_set": ""}, {"tag": "hours", "patterns": ["when are you guys open", "what are your hours", "hours of operation"], "responses": ["We are open 7am-4pm Monday-Friday!"], "context_set": ""}, {"tag": "diabetes0", "patterns": ["there are how many types of diabetes"], "responses": ["There are three main types of diabetes   type 1, type 2 and gestational. Type 1 diabetes can develop at any age, but occurs most frequently in children and adolescents."], "context_set": ""}, {"tag": "diabetes1", "patterns": ["can a child have diabetes"], "responses": ["Type 2 diabetes in children is a chronic disease that affects the way your child's body processes sugar (glucose). Without treatment, the disorder causes sugar to build up in the blood, which can lead to serious long-term consequences. Type 2 diabetes occurs more commonly in adults."], "context_set": ""}, {"tag": "diabetes2", "patterns": ["is it possible to control diabetes without medicine"], "responses": ["Although there's no cure for type 2 diabetes, studies show it's possible for some people to reverse it. Through diet changes and weight loss, you may be able to reach and hold normal blood sugar levels without medication. This doesn't mean you're completely cured."], "context_set": ""}, {"tag": "diabetes3", "patterns": ["What should my blood sugar be?"], "responses": ["The American Diabetes Association recommends a blood glucose range of 80-130 before meals and less than 180 about 2 hours after a meal. This range should place your A1c under 7."], "context_set": ""}, {"tag": "diabetes4", "patterns": ["What is an A1c?"], "responses": [" A Hemoglobin A1c is a 2-3 month average of your blood sugars. This result gives you a good idea of how well your diabetes is being managed/controlled. The American Diabetes Association recommends an A1c of less than 7 to keep the risk of complications low."], "context_set": ""}, {"tag": "diabetes5", "patterns": ["What can I eat if I have diabetes?"], "responses": ["You can eat just about anything you want. It is about knowing proper portion sizes and how much you are putting on your plate. A dietitian can help you learn to count carbohydrates and with meal planning that is specific for you."], "context_set": ""}, {"tag": "diabetes6", "patterns": ["Why does it matter if my blood sugar is 120 or 200?"], "responses": ["It is very important to keep your blood sugar level under control. When your blood sugar level is high, it can cause damage in your veins and arteries. This damage could lead to complications later such as heart attacks, strokes, kidney disease, neuropathies, vision problems, etc. "], "context_set": ""}, {"tag": "diabetes7", "patterns": ["What foods have carbohydrates?"], "responses": ["Fruits, starchy vegetables, milk, yogurt, rice, cereals, bread and other grains all have carbs and give you important nutrients. Many snack foods, such as pretzels, chips and popcorn, have carbs. Sweets, including regular soda, cakes, candy and cookies, also contain carbohydrates. Be sure to check the nutrition label on each food item to determine carbohyrdate content."], "context_set": ""}, {"tag": "diabetes8", "patterns": ["Do I need to follow a low carb diet?"], "responses": ["Carbohydrates are an important part of a healthy diet. With diabetes, watching portion sizes and getting most of your carbs from fruits, vegetables, whole grains and low fat milk and yogurt is key. Besides counting carbs, people with diabetes can also benefit from eating lower fat, high fiber foods and just enough calories to maintain a healthy weight."], "context_set": ""}, {"tag": "diabetes9", "patterns": ["If it is sugar free, I can eat as much as I want, right?"], "responses": ["ugar free foods can be part of a healthy meal plan in small amounts. Keep in mind though that some of these foods still have carbs (in the form of other sweeteners such as sorbitol, isomalt, and mannitol) and may affect your blood glucose levels. Many sugar free foods have calories and carbohydrates and lots of fat. Make sure you read the nutrition labels."], "context_set": ""}, {"tag": "diabetes10", "patterns": ["What resources are available to help me control my diabetes?  "], "responses": ["Diabetes classes are offered by Maury Regional Health and can be scheduled by your provider at with a certified diabetes educator and/or dietitian. This class is recognized by the American Diabetes Association."], "context_set": ""}, {"tag": "diabetes11", "patterns": ["When do you check your glucose?"], "responses": ["The common times for checking your blood sugar are when you first wake up (fasting), before a meal, 2 hours after a meal, and at bedtime. Talk with your health care team about what times are best for you to check your blood sugar."], "context_set": ""}, {"tag": "diabetes12", "patterns": ["Diabetes symptoms?"], "responses": ["Urinate (pee) a lot, often at night , Are very thirsty , Lose weight without trying , Are very hungry , Have blurry vision , Have numb or tingling hands or feet , Feel very tired , Have very dry skin , Have sores that heal slowly , Have more infections than usual "], "context_set": ""}, {"tag": "diabetes13", "patterns": ["Symptoms of Type 1 Diabetes"], "responses": ["People who have type 1 diabetes may also have nausea, vomiting, or stomach pains. Type 1 diabetes symptoms can develop in just a few weeks or months and can be severe. Type 1 diabetes usually starts when you re a child, teen, or young adult but can happen at any age."], "context_set": ""}, {"tag": "diabetes14", "patterns": ["Symptoms of Type 2 Diabetes"], "responses": ["Type 2 diabetes symptoms often take several years to develop. Some people don t notice any symptoms at all. Type 2 diabetes usually starts when you re an adult, though more and more children and teens are developing it. Because symptoms are hard to spot, it s important to know the risk factors for type 2 diabetes. Make sure to visit your doctor if you have any of them."], "context_set": ""}, {"tag": "diabetes15", "patterns": ["Symptoms of Gestational Diabetes"], "responses": ["Gestational diabetes (diabetes during pregnancy) usually doesn t have any symptoms. If you re pregnant, your doctor should test you for gestational diabetes between 24 and 28 weeks of pregnancy. If needed, you can make changes to protect your health and your baby s health."], "context_set": ""}, {"tag": "diabetes16", "patterns": ["Medications for type 1 diabetes"], "responses": ["Insulin , Short-acting insulin , Rapid-acting insulins , Intermediate-acting insulin , Long-acting insulins , Combination insulins"], "context_set": ""}, {"tag": "diabetes17", "patterns": ["if i get diabetes will i die"], "responses": ["Diabetes can lead to serious complications which can affect many different parts of your body. In the worst cases, diabetes can kill you. Each week diabetes causes thousands of complications like stroke, amputation, kidney failure, heart attack and heart failure."], "context_set": ""}, {"tag": "diabetes18", "patterns": ["Can diabetics live a long life?"], "responses": ["Type 2 diabetes is a serious condition that can lead to life-threatening complications. However, by adopting effective management strategies, there is a good chance that many people with type 2 diabetes can expect to live as long as a person without the condition."], "context_set": ""}, {"tag": "diabetes19", "patterns": ["Can 20 year olds get diabetes?"], "responses": ["People can develop type 1 diabetes at any age, from early childhood to adulthood, but the average age at diagnosis is 13 years. An estimated 85% of all type 1 diagnoses take place in people aged under 20 years."], "context_set": ""}, {"tag": "diabetes20", "patterns": ["Who is most at risk of diabetes?"], "responses": ["are overweight or obese. , are age 45 or older. , have a family history of diabetes. , are African American, Alaska Native, American Indian, Asian American, Hispanic/Latino, Native Hawaiian, or Pacific Islander. , have high blood pressure"], "context_set": ""}, {"tag": "diabetes21", "patterns": ["Can diabetics drink alcohol?"], "responses": ["Drink in Moderation"], "context_set": ""}, {"tag": "diabetes22", "patterns": ["What is the youngest age to get diabetes?"], "responses": ["It used to be called juvenile diabetes because most of the people who got it were young children. Your child could get type 1 diabetes as an infant, or later, as a toddler or a teen. Most often, it appears after age 5. But some people don't get it until their late 30s."], "context_set": ""}, {"tag": "diabetes23", "patterns": ["Can eating too much sugar cause diabetes?"], "responses": ["Excessive amounts of added sugars have been associated with an increased risk of type 2 diabetes, likely due to negative effects on the liver and a higher risk of obesity. Natural sugars like those found in fruits and vegetables are not linked to diabetes risk   whereas artificial sweeteners are."], "context_set": ""}, {"tag": "diabetes24", "patterns": ["Can diabetics eat pizza?"], "responses": ["Pizza may actually be a good choice for people with type 2 diabetes; just be sure to order the thin-crust type and top it with vegetables rather than high-fat meats and extra cheese. It's also a good idea to watch portion sizes."], "context_set": ""}, {"tag": "diabetes25", "patterns": ["What diet causes diabetes?"], "responses": ["A diet high in fat, calories, and cholesterol increases your risk of diabetes. A poor diet can lead to obesity (another risk factor for diabetes) and other health problems. A healthy diet is high in fiber and low in fat, cholesterol, salt, and sugar."], "context_set": ""}, {"tag": "diabetes26", "patterns": ["Does exercise reverse diabetes?"], "responses": ["Exercise helps the body to become more sensitive to its insulin. In combination with a healthy diet, exercise can reduce the demand for insulin in the body and therefore help reverse diabetes."], "context_set": ""}, {"tag": "diabetes27", "patterns": ["Is 200 blood sugar normal after eating?"], "responses": ["Less than 140 mg/dL (7.8 mmol/L) is normal. 140 to 199 mg/dL (7.8 mmol/L and 11.0 mmol/L) is diagnosed as prediabetes. 200 mg/dL (11.1 mmol/L) or higher after two hours suggests diabetes."], "context_set": ""}, {"tag": "diabetes28", "patterns": ["Is 216 blood sugar high?"], "responses": ["fasting blood sugar of 126 mg/dL or higher = diabetes. fasting blood sugar of 100 to 125 mg/dL = prediabetes. fasting blood sugar less than 100 mg/dL = normal."], "context_set": ""}, {"tag": "diabetes29", "patterns": ["What happens when your blood sugar is 300?"], "responses": ["A reading above 300 mg/dL can be dangerous, according to the University of Michigan, which recommends immediately informing your doctor if you have two or more readings of 300 mg/dL in a row. In severe cases, very high blood sugar levels (well above 300 mg/dL) can result in coma."], "context_set": ""}, {"tag": "diabetes30", "patterns": ["At what sugar level is diabetic coma?"], "responses": ["A diabetic coma could happen when your blood sugar gets too high -- 600 milligrams per deciliter (mg/dL) or more -- causing you to become very dehydrated. It usually affects people with type 2 diabetes that isn't well-controlled."], "context_set": ""}, {"tag": "diabetes31", "patterns": ["When should a diabetic go to hospital?"], "responses": ["You should call your doctor if you have high blood sugar levels throughout the day, if you find your blood sugar level is always high at the same time each day, or if you are having symptoms of high blood sugar like drinking or urinating (peeing) a lot more than normal."], "context_set": ""}, {"tag": "diabetes32", "patterns": ["How long does a diabetic patient live?"], "responses": ["The combined diabetic life expectancy is 74.64 years comparable to the life expectancy in the general population."], "context_set": ""}, {"tag": "diabetes33", "patterns": ["Can a 23 year old get diabetes?"], "responses": ["It might seem surprising that someone so young could develop type 2 diabetes, but the disease is on the rise among the under-30 set. In fact, 5.7 percent of all new cases of diabetes occur in people between 18 and 29, the U.S. Centers for Disease Control and Prevention estimates."], "context_set": ""}, {"tag": "diabetes34", "patterns": ["Can diabetes start suddenly?"], "responses": ["Over time, as the insulin in your body is sapped with no new supply being produced to replace it, symptoms begin to appear and accelerate. When you reach a point where there's no insulin and too much accumulated glucose in your bloodstream, type 1 diabetes symptoms develop rapidly and have to be addressed immediately."], "context_set": ""}, {"tag": "diabetes35", "patterns": ["What does sugar in urine look like?"], "responses": ["The sugar is then excreted in your urine. The excess sugar can make it appear cloudy or even smell sweet or fruity. For some people, this is the first sign of diabetes. If you suddenly notice cloudy urine that smells sweet, see a doctor right away."], "context_set": ""}, {"tag": "diabetes36", "patterns": ["How often do you pee diabetes?"], "responses": ["People with diabetes tend to urinate a lot more than the average person - who normally urinates four to seven times in 24 hours. For someone who doesn't have diabetes, the body reabsorbs glucose as it passes through the kidneys"], "context_set": ""}, {"tag": "diabetes37", "patterns": ["Is drinking water good for diabetes?"], "responses": ["When it comes to hydration, water is the best option for people with diabetes. That's because it won't raise your blood sugar levels. High blood sugar levels can cause dehydration. Drinking enough water can help your body eliminate excess glucose through urine."], "context_set": ""}, {"tag": "diabetes38", "patterns": ["Can diabetes be cured by exercise?"], "responses": ["People with Type 2 diabetes can reverse their condition with diet and exercise, although remission is not very common, according to a new study from the Centers for Disease Control and Prevention."], "context_set": ""}, {"tag": "diabetes39", "patterns": ["Is jogging good for diabetes?"], "responses": ["Running can be an ideal form of exercise for people with diabetes as it helps improve the body's sensitivity to insulin. This can be especially useful for people with type 2 diabetes to help combat insulin resistance."], "context_set": ""}, {"tag": "diabetes40", "patterns": ["What is the best time to exercise for diabetics?"], "responses": ["Those with type 2 diabetes are supposed to keep levels at 160 mg/dl within two hours of a meal. Because exercising reduces blood glucose concentrations, it's a good idea to start exercising about 30 minutes after the beginning of a meal, researchers concluded."], "context_set": ""}, {"tag": "diabetes41", "patterns": ["Does cycling reduce diabetes?"], "responses": ["The study shows that those who start cycling late (on average) get a 20% reduction in the risk of diabetes. The beneficial effects of two-wheelers emerge independently of other factors that may affect the risk of disease, such as nutrition, weight problems."], "context_set": ""}, {"tag": "diabetes42", "patterns": ["How much should diabetics walk?"], "responses": ["Benefits. By walking every day for 30 minutes to an hour, people with diabetes can reap the following benefits: Improved glucose control.  Exercise helps muscles absorb blood sugar, preventing it from building up in the bloodstream."], "context_set": ""}, {"tag": "diabetes43", "patterns": ["How do you bring your blood sugar down quickly?"], "responses": ["When your blood sugar level gets too high   known as hyperglycemia or high blood glucose   the quickest way to reduce it is to take fast-acting insulin. Exercising is another fast, effective way to lower blood sugar."], "context_set": ""}, {"tag": "diabetes44", "patterns": ["What foods help get rid of diabetes?"], "responses": ["Non-Starchy Vegetables. Non-starchy vegetables are one of the best foods you can eat as a diabetic.,Leafy Greens.,Fatty Fish.,Nuts and Eggs.,Seeds.,Natural Fats.,Apple Cider

Vinegar.,Cinnamon and Turmeric."], "context_set": ""}, {"tag": "diabetes45", "patterns": ["Can walking 30 minutes a day lower blood sugar?"], "responses": ["Good news: Two new studies found that exercising 30 minutes a day reduces your risk of diabetes by 25 percent, and walking for 10 minutes after meals lowers your blood sugar by 22 percent."], "context_set": ""}, {"tag": "diabetes46", "patterns": ["Does turmeric lower blood sugar?"], "responses": ["Turmeric may help manage high glucose levels in your blood. The spice was shown to increase insulin sensitivity, which leads to lower blood sugar levels."], "context_set": ""}, {"tag": "diabetes47", "patterns": ["Is rice good for diabetes?"], "responses": ["Rice is rich in carbohydrates and can have a high GI score. If you have diabetes, you may think that you need to skip it at dinner, but this isn't always the case. You can still eat rice if you have diabetes. You should avoid eating it in large portions or too frequently, though."], "context_set": ""}, {"tag": "diabetes48", "patterns": ["Will boiled eggs lower blood sugar?"], "responses": ["A hard-boiled egg is a handy high-protein snack if you have diabetes. The protein will help keep you full without affecting your blood sugar. Protein not only slows digestion, it also slows glucose absorption. This is very helpful if you have diabetes."], "context_set": ""}, {"tag": "diabetes49", "patterns": ["Is Potato bad for diabetes?"], "responses": ["Eating too many potatoes can present problems for blood sugar control in people with diabetes. However, potatoes are a good source of vitamins, minerals, and fiber, and people with diabetes can enjoy them as part of a healthful diet."], "context_set": ""}, {"tag": "diabetes50", "patterns": ["Can we eat Maida in diabetes?"], "responses": ["Over consumption of maida is linked with constipation. The glycemic index of maida is very high. It is not suitable to patients suffering from diabetes and obesity."], "context_set": ""}, {"tag": "diabetes51", "patterns": ["Is bread bad for diabetics?"], "responses": ["Contrary to popular belief, people with type 2 diabetes can, in fact, eat bread   the right kinds, in moderation. The American Diabetes Association (ADA) puts it this way: Starchy foods can be part of a healthy meal plan, but portion size is key."], "context_set": ""}, {"tag": "diabetes52", "patterns": ["Is Chicken Good for diabetes?"], "responses": ["Chicken can be a great option for people with diabetes. All cuts of chicken are high in protein and many are low in fat. When prepared in a healthy way, chicken can be a great ingredient in a healthy diabetic eating plan."], "context_set": ""}, {"tag": "diabetes53", "patterns": ["Can diabetics eat carrots?"], "responses": ["Uncontrolled blood sugar levels can lead to type 2 diabetes or worsen your disease. Carrots can be a safe choice if you have diabetes and are watching your blood sugar levels. They're also non-starchy vegetables. So you can even enjoy small amounts of carrots if you're following the ketogenic, or keto, diet."], "context_set": ""}, {"tag": "diabetes54", "patterns": ["Can a diabetic eat chips?"], "responses": ["You may love their lip-smackin' saltiness, but potato chips, tortilla chips, or corn chips (including those in restaurant nachos), crackers, and pretzels are not the best food choices for people living with diabetes."], "context_set": ""}, {"tag": "diabetes55", "patterns": ["Is papaya good for diabetes?"], "responses": ["Papaya isn't only a good choice for people with diabetes because of its medium GI. Eating papaya might also lower your blood sugar. According to some reports , papaya may have a hypoglycemic effect on the body. The fruit contains flavonoids, which are natural antioxidants that may help regulate blood sugar."], "context_set": ""}, {"tag": "diabetes56", "patterns": ["Can diabetic patient eat Maggi?"], "responses": ["Eating maggie is not a good option as it is loaded with refined carbs, bad fats, preservatives. It wont just affect your sugar at that moment but will have long term effects also"], "context_set": ""}, {"tag": "diabetes57", "patterns": ["Can diabetics eat butter?"], "responses": ["Eating butter in moderation is safe for those with diabetes. Choosing real butter instead of margarine will decrease trans fat intake and have a better overall impact on heart health and diabetes management. Since butter is a saturated fat, being mindful of total daily intake is important"], "context_set": ""}, {"tag": "diabetes58", "patterns": ["Can diabetics drink lemonade?"], "responses": ["Your best bet is to make lemonade at home. Mix water, fresh-squeezed lemons, zero-calorie sweetener, and ice for a truly refreshing beverage without a single carb or calorie in sight."], "context_set": ""}, {"tag": "diabetes59", "patterns": ["Which biscuit is good for diabetes?"], "responses": ["Nutrichoice Essentials are diabetic-friendly biscuits from Britannia. They are available in Oats and Ragi variants in On the Go Convenient packs. They *help manage blood sugar levels and are diabetic friendly as they have: High dietary fibre."], "context_set": ""}, {"tag": "diabetes60", "patterns": ["Is yogurt OK for diabetics?"], "responses": ["Yogurts that contain a total carbohydrate content of 15 g or less per serving are ideal for people with diabetes. Look for yogurts that are high in protein and low in carbohydrates, such as unflavored Greek yogurt."], "context_set": ""}, {"tag": "diabetes61", "patterns": ["Can diabetics eat ice cream?"], "responses": ["DIABETICS, YOU CAN EAT ICE CREAM AND THIS IS THE BEST WAY TO EAT IT: People who have diabetes usually think ice cream is off limits for them. While it is true since most varieties available in the market are high in sugar, which can raise your blood sugar levels rapidly"], "context_set": ""}, {"tag": "diabetes62", "patterns": ["Is coconut water good for diabetes?"], "responses": ["Coconut water is packed with nutrients and minerals along with having a low glycemic index. What's more, it is delicious and absolutely natural! Due to all these reasons, it is often recommended for people with diabetes."], "context_set": ""}, {"tag": "diabetes63", "patterns": ["Are dates good for diabetes?"], "responses": ["Dates have a low GI, which means they're less likely to spike your blood sugar levels, making them a safe choice for people with diabetes. Moreover, dates have a medium GL, which means that 1 or 2 fruits at a time are a good choice"], "context_set": ""}, {"tag": "diabetes64", "patterns": ["Is idli good for diabetic?"], "responses": ["Fibre-rich, high in nutritional value and with a low glycemic index, Oats can be eaten boiled with milk and with vegetables. Oats dosas, idlis and pancakes work well with the Indian palate. Eating them regularly has known to decrease insulin resistance in diabetics."], "context_set": ""}, {"tag": "diabetes65", "patterns": ["s Almond good for diabetics?"], "responses": ["Almonds are a good mid time snack for a diabetic patient. The presence of magnesium makes it beneficial for diabetesand controls blood sugar levels. Few studies have been conducted recently which indicates that almonds if consumed in a good quantity for long run can help controlling blood sugars."], "context_set": ""}, {"tag": "diabetes66", "patterns": ["How much water should a diabetic drink daily?"], "responses": ["If you're living with diabetes, you should drink plenty of fluids   about 1.6 liters (L) or 6.5 cups per day for women; and 2 L or 8.5 glasses per day for men."], "context_set": ""}, {"tag": "diabetes67", "patterns": ["What drink lowers blood sugar?"], "responses": ["Pomegranate juice is said to help lower blood sugar levels within 15 minutes of drinking it. A study examined the effects one drink of the juice could have on a diabetic and found it changed the levels of their blood sugar - in a good way."], "context_set": ""}, {"tag": "diabetes68", "patterns": ["Are ginger biscuits good for diabetics?"], "responses": ["Ginger can be an effective addition to your diabetes treatment if you use it in moderation. Eating up to 4 grams per day may help lower your blood sugar levels and regulate insulin production. Be sure to talk with your doctor before adding this to your treatment regimen."], "context_set": ""}, {"tag": "diabetes69", "patterns": ["Can diabetes be cured permanently by Yoga?"], "responses": ["However, diabetes can be controlled to such an extent that you may not need medication but it cannot be cured, even through yoga."], "context_set": ""}, {"tag": "diabetes70", "patterns": ["Which yoga is best for diabetes?"], "responses": ["Walking. Cycling. Swimming. Team sports. Aerobic dance. Weightlifting. Resistance band exercises. Calisthenics."], "context_set": ""}, {"tag": "diabetes71", "patterns": ["Can Ramdev cure diabetes?"], "responses": ["According to Swami Ramdev, diabetes can be easily overcome by doing 5 pranayamas and 5 yoga asanas daily. These yoga tips will help get rid of all types of diabetes by staying at home. Along with these Yogasanas, drink a kada (decoction) of Giloy, Basil, Pepper and Ashwagandha."], "context_set": ""}, {"tag": "diabetes72", "patterns": ["Will surya namaskar reduce diabetes?"], "responses": ["Surya Namaskar (sun salutations)Surya Namaskar is an extremely beneficial yoga practice for people suffering from diabetes as it improves the blood circulation and the management of insulin in the body. If performed at a slow pace (six rounds a minute), these asanas work best to give you major benefits."], "context_set": ""}, {"tag": "diabetes73", "patterns": ["Does yoga increase blood sugar?"], "responses": ["Exercising the muscles   Like other forms of exercise, yoga increases glucose uptake by muscular cells, which in turn, helps to lower blood sugar levels, improve circulation and reduce the risk of cardiovascular disease."], "context_set": ""}, {"tag": "diabetes74", "patterns": ["What does Ayurveda say about diabetes?"], "responses": ["According to Ayurveda, a body is in its healthiest state if and when Vata, Pitta and Kapha are in balance. Any imbalance will lead to health ailments. Diabetes is essentially cause by the imbalance of Pitta in the body."], "context_set": ""}, {"tag": "diabetes75", "patterns": ["How can I improve my pancreas naturally?"], "responses": ["To get your pancreas healthy, focus on foods that are rich in protein, low in animal fats, and contain antioxidants. Try lean meats, beans and lentils, clear soups, and dairy alternatives (such as flax milk and almond milk). Your pancreas won't have to work as hard to process these."], "context_set": ""}, {"tag": "diabetes76", "patterns": ["Can yoga cure Type 1 diabetes?"], "responses": ["Yoga not only increases blood supply throughout the body, improving insulin administration but also builds muscles and reduces stress; key things that are essential if you live with type 1 diabetes."], "context_set": ""}, {"tag": "diabetes77", "patterns": ["Can yoga reverse Type 2 diabetes?"], "responses": ["The results of one 2016 review found that yogic practices can significantly help to manage type 2 diabetes. Researchers concluded that yoga had a positive improvement on blood sugar levels, lipid levels, and body composition."], "context_set": ""}, {"tag": "diabetes78", "patterns": ["Which Pranayama is suitable for diabetes?"], "responses": ["One of the basic preparations for Pranayama is Nadi Shodhan Pranayama or alternate nostril breathing, this type is found useful in diabetes as Alternate nostril breathing has calming effect on nervous system, which reduces stress levels, helping in diabetes treatment."], "context_set": ""}, {"tag": "diabetes79", "patterns": ["How can yoga activate the pancreas naturally?"], "responses": ["Studies have also confirmed that practising certain asanas such as Ardha Matsyendrasana (half-twist pose) combined with Dhanurasana (bow pose), Vakrasana (twisted pose), Matsyendrasana (half-spinal twist), Halasana (plough pose) squeezes and compresses the abdomen and helps stimulate the pancreatic secretions "], "context_set": ""}, {"tag": "diabetes80", "patterns": ["What exercises improve pancreas?"], "responses": ["Short, functional-movement and resistance training workouts, called functional high-intensity training (F-HIT), may improve beta-cell function in adults with type 2 diabetes, new research shows. Beta cells in the pancreas produce, store and secrete insulin, which allows your body to use sugar for energy."], "context_set": ""}, {"tag": "diabetes81", "patterns": ["Which yoga pose is good for pancreas?"], "responses": ["One yoga practice specific to the pancreas is Uddiyana Bandha or the Stomach Lock Pose. This pose puts pressure into the pancreatic area squeezing it tightly."], "context_set": ""}, {"tag": "diabetes82", "patterns": ["Is walking good for pancreas?"], "responses": ["However, we found that the group that practiced walking showed normalization of the endocrine part of the pancreas. Few studies have reported the beneficial effects of physical training on the pancreas alone."], "context_set": ""}, {"tag": "diabetes83", "patterns": ["Is fasting good for the pancreas?"], "responses": ["Fasting is known to cause physiological changes in the endocrine pancreas, including decreased insulin secretion and increased reactive oxygen species (ROS) production."], "context_set": ""}, {"tag": "diabetes84", "patterns": ["How do I know if my pancreas is producing insulin?"], "responses": ["C-peptide and insulin are released from the pancreas at the same time and in about equal amounts. So a C-peptide test can show how much insulin your body is making. This test can be a good way to measure insulin levels because C-peptide tends to stay in the body longer than insulin."], "context_set": ""}, {"tag": "diabetes85", "patterns": ["how to prevent diabetes?"], "responses": ["Lose extra weight. Losing weight reduces the risk of diabetes.,Be more physically active. There are many benefits to regular physical activity.,Eat healthy plant foods. Plants provide vitamins, minerals and carbohydrates in your diet"], "context_set": ""}, {"tag": "diabetes86", "patterns": ["Why do diabetics get so angry?"], "responses": ["Since the hormones that regulate blood sugar also regulate stress levels, when your blood sugar is off, you can become enraged or depressed, which in turn makes it harder to regulate your blood sugar."], "context_set": ""}, {"tag": "diabetes87", "patterns": ["Does diabetes affect memory?"], "responses": ["Uncontrolled diabetes may increase the risk of experiencing cognitive problems, such as memory loss. Higher than normal blood glucose levels can damage nerve cells, supportive glial cells, and blood vessels in both peripheral nerves of the body and the brain."], "context_set": ""}, {"tag": "diabetes88", "patterns": ["How can diabetes affect you emotionally?"], "responses": ["The fear of blood sugar fluctuations can be very stressful. Changes in blood sugar can cause rapid changes in mood and other mental symptoms such as fatigue, trouble thinking clearly, and anxiety. Having diabetes can cause a condition called diabetes distress which shares some traits of stress, depression and anxiety."], "context_set": ""}, {"tag": "diabetes89", "patterns": ["Can diabetes cause strange behavior?"], "responses": ["In diabetes, irrational behavior happens because glucose levels that are too high (hyperglycemia) or, especially, too low (hypoglycemia) impede self-control. When people lack their normal level of self-control, they often: are impulsive. disregard long-term consequences of their actions."], "context_set": ""}, {"tag": "diabetes90", "patterns": ["is a hereditary disease?"], "responses": ["Diabetes is a hereditary disease, which means that the child is at high risk of developing diabetes compared to the general population at the given age. Diabetes can be inherited from either mother or father."], "context_set": ""}, {"tag": "diabetes91", "patterns": ["Which is hereditary type 1 or type 2 diabetes?"], "responses": ["Type 2 diabetes has a stronger link to family history and lineage than type 1, and studies of twins have shown that genetics play a very strong role in the development of type 2 diabetes"], "context_set": ""}, {"tag": "diabetes92", "patterns": ["Can you avoid diabetes if it runs in your family?"], "responses": ["Even if you have a family health history of diabetes, you can prevent or delay type 2 diabetes by eating healthier, being physically active, and maintaining or reaching a healthy weight. This is especially important if you have prediabetes, and taking these steps can reverse prediabetes."], "context_set": ""}, {"tag": "diabetes93", "patterns": ["Can I get diabetes if my grandma has it?"], "responses": ["If there is a history of a type of diabetes in a person's family, they may have a higher risk of developing the same condition. Genetic factors can make some people more vulnerable to some types of diabetes. However, a person may not inherit the condition, and there may be ways to reduce the risk."], "context_set": ""}, {"tag": "diabetes94", "patterns": ["Why do people get diabetes?"], "responses": ["Diabetes is a chronic disease that occurs because the body is unable to use blood sugar (glucose) properly. The exact cause of this malfunction is unknown, but genetic and environmental factors play a part. Risk factors for diabetes include obesity and high levels of cholesterol."], "context_set": ""}, {"tag": "diabetes95", "patterns": ["Does coffee affect blood sugar?"], "responses": ["For most young, healthy adults, caffeine doesn't appear to noticeably affect blood sugar (glucose) levels, and having up to 400 milligrams a day appears to be safe."], "context_set": ""}, {"tag": "diabetes96", "patterns": ["Is milk good for a diabetic?"], "responses": ["Milk Nutrition. Dairy is important for your diet because it's an excellent source of calcium. But it may also be high in fat and carbs, making it risky for people with diabetes."], "context_set": ""}, {"tag": "diabetes97", "patterns": ["What is the miracle fruit that cures diabetes?"], "responses": ["MiraBurst is particularly beneficial for diabetics and borderline diabetics. MiraBurst can help diabetics and pre-diabetics improve their body's sensitivity to their own insulin and manage blood sugar levels."], "context_set": ""}, {"tag": "diabetes98", "patterns": ["Which is worse type 1 or 2 diabetes?"], "responses": ["Type 2 diabetes is often milder than type 1. But it can still cause major health complications, especially in the tiny blood vessels in your kidneys, nerves, and eyes. Type 2 also raises your risk of heart disease and stroke."], "context_set": ""}, {"tag": "diabetes99", "patterns": ["is it possible to get rid of diabetes"], "responses": ["There is no known cure for type 2 diabetes. But it can be controlled. And in some cases, it goes into remission. For some people, a diabetes-healthy lifestyle is enough to control their blood sugar levels."], "context_set": ""}]}

## 2. Preprocess data

When working with text data, we need to perform various preprocessing on the data before design an ANN model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list. We also create a list of classes for our tags.

Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

## 3. **Create training and testing data**

Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn’t understand text so we will convert text into numbers.

## 4.Build the model

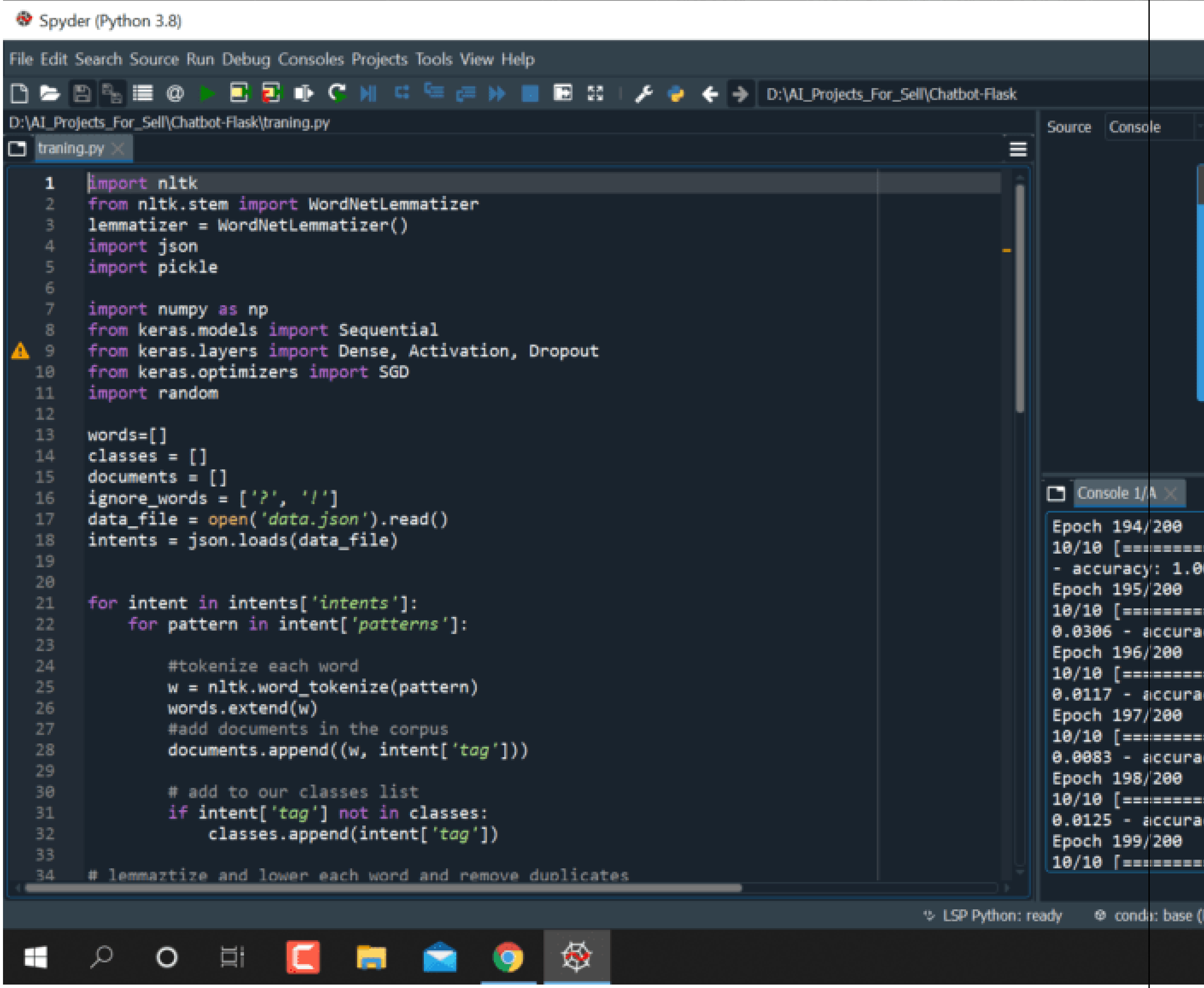
We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as ‘model.h5’.

**Training.py is shown below**

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('data.json').read()
intents = json.loads(data_file)
for intent in intents['intents']:
for pattern in intent['patterns']:
#tokenize each word
w = nltk.word_tokenize(pattern)
words.extend(w)
#add documents in the corpus
documents.append((w, intent['tag']))
# add to our classes list
if intent['tag'] not in classes:
classes.append(intent['tag'])
# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
```

```
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('texts.pkl','wb'))
pickle.dump(classes,open('labels.pkl','wb'))
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
# initialize our bag of words
bag = []
# list of tokenized words for the pattern
pattern_words = doc[0]
# lemmatize each word - create base word, in attempt to represent related words
pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
# create our bag of words array with 1, if word match found in current pattern
for w in words:
bag.append(1) if w in pattern_words else bag.append(0)
# output is a '0' for each tag and '1' for current tag (for each pattern)
output_row = list(output_empty)
output_row[classes.index(doc[1])] = 1
training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer
contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
```

```
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save('model.h5', hist)
print("model created")
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\AI_Projects_For_Sell\Chatbot-Flask
D:\AI_Projects_For_Sell\Chatbot-Flask\traning.py
traning.py
Source Console
Console 1/A
LSP Python: ready
conda: base (

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('data.json').read()
intents = json.loads(data_file)


for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates
```

```
Epoch 194/200
10/10 [========
- accuracy: 1.0
Epoch 195/200
10/10 [========
0.0306 - accura
Epoch 196/200
10/10 [========
0.0117 - accura
Epoch 197/200
10/10 [========
0.0083 - accura
Epoch 198/200
10/10 [========
0.0125 - accura
Epoch 199/200
10/10 [========
```

Training-ANN Model

## 5.Predict the response (Flask web-based GUI)

Now to predict the sentences and get a response from the user to let us create a new file ‘app.py’using flask web-based framework.

**Note for making flask app we need to make to folders name as static and templates and app.py files.**

- **static folder contains a subfolder with name styles. The styles folder contain css file with the** name **style.css**
- ***Templates folder HTML file with the name of index.html***
- **app.py file for run the flask app using IDE.**

**static/styles/style.css**

```
:root {
--body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
--msger-bg: #fff;
--border: 2px solid #ddd;
--left-msg-bg: #ececec;
--right-msg-bg: #579ffb;
}
html {
box-sizing: border-box;
}
*,
*:before,
*:after {
margin: 0;
padding: 0;
box-sizing: inherit;
}
body {
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
background-image: var(--body-bg);
font-family: Helvetica, sans-serif;
}
.msger {
display: flex;
flex-flow: column wrap;
justify-content: space-between;
width: 100%;
```

```
max-width: 867px;
margin: 25px 10px;
height: calc(100% - 50px);
border: var(--border);
border-radius: 5px;
background: var(--msger-bg);
box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);
}
.msger-header {
/* display: flex; */
font-size: medium;
justify-content: space-between;
padding: 10px;
text-align: center;
border-bottom: var(--border);
background: #eee;
color: #666;
}
.msger-chat {
flex: 1;
overflow-y: auto;
padding: 10px;
}
.msger-chat::-webkit-scrollbar {
width: 6px;
}
.msger-chat::-webkit-scrollbar-track {
background: #ddd;
}
.msger-chat::-webkit-scrollbar-thumb {
background: #bdbdbd;
}
.msg {
display: flex;
align-items: flex-end;
margin-bottom: 10px;
}
.msg-img {
width: 50px;
```

```
height: 50px;
margin-right: 10px;
background: #ddd;
background-repeat: no-repeat;
background-position: center;
background-size: cover;
border-radius: 50%;
}
.msg-bubble {
max-width: 450px;
padding: 15px;
border-radius: 15px;
background: var(--left-msg-bg);
}
.msg-info {
display: flex;
justify-content: space-between;
align-items: center;
margin-bottom: 10px;
}
.msg-info-name {
margin-right: 10px;
font-weight: bold;
}
.msg-info-time {
font-size: 0.85em;
}
.left-msg .msg-bubble {
border-bottom-left-radius: 0;
}
.right-msg {
flex-direction: row-reverse;
}
.right-msg .msg-bubble {
background: var(--right-msg-bg);
color: #fff;
border-bottom-right-radius: 0;
}
.right-msg .msg-img {
```

```
margin: 0 0 0 10px;
}
.msger-inputarea {
display: flex;
padding: 10px;
border-top: var(--border);
background: #eee;
}
.msger-inputarea * {
padding: 10px;
border: none;
border-radius: 3px;
font-size: 1em;
}
.msger-input {
flex: 1;
background: #ddd;
}
.msger-send-btn {
margin-left: 10px;
background: rgb(0, 196, 65);
color: #fff;
font-weight: bold;
cursor: pointer;
transition: background 0.23s;
}
.msger-send-btn:hover {
background: rgb(0, 180, 50);
}
.msger-chat {
background-color: #fcfcfe;
background-image: url("data:image/svg+xml,%3Csvg xmlns='http://www.w3.org/2000/svg'
width='260' height='260' viewBox='0 0 260 260'%3E%3Cg fill-rule='evenodd'%3E%3Cg
fill='%23dddddd' fill-opacity='0.4'%3E%3Cpath d='M24.37 16c.2.65.39 1.32.54 2H21.17l1.17
2.34.45.9-.24.11V28a5 5 0 0 1-2.23 8.94l-.02.06a8 8 0 0 1-7.75 6h-20a8 8 0 0 1-7.74-6l-.02-
.06A5 5 0 0 1-17.45 28v-6.76l-.79-1.58-.44-.9.9-.44.63-.32H-20a23.01 23.01 0 0 1 44.37-2zm-
36.82 2a1 1 0 0 0-.44.1l-3.1 1.56.89 1.79 1.31-.66a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-
1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0
.86.02l2.88-1.27a3 3 0 0 1 2.43 0l2.88 1.27a1 1 0 0 0 .85-.02l3.1-1.55-.89-1.79-1.42.71a3 3 0 0
```

```
1-2.56.06l-2.77-1.23a1 1 0 0 0-.4-.09h-.01a1 1 0 0 0-.4.09l-2.78 1.23a3 3 0 0 1-2.56-.06l-2.3-
1.15a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.1L.9 19.22a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-
.11h-.01a1 1 0 0 0-.44.1l-2.21 1.11a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01zm0-2h-
4.9a21.01 21.01 0 0 1 39.61 0h-2.09l-.06-.13-.26.13h-32.31zm30.35 7.68l1.36-.68h1.3v2h-36v-
1.15l.34-.17 1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69
0L2.26 23h2.59l1.36.68a3 3 0 0 0 2.56.06l1.67-.74h3.23l1.67.74a3 3 0 0 0 2.56-.06zM-13.82
27l16.37 4.91L18.93 27h-32.75zm-.63 2h.34l16.66 5 16.67-5h.33a3 3 0 1 1 0 6h-34a3 3 0 1 1
0-6zm1.35 8a6 6 0 0 0 5.65 4h20a6 6 0 0 0 5.66-4H-13.1z'/%3E%3Cpath id='path6_fill-copy'
d='M284.37 16c.2.65.39 1.32.54 2H281.17l1.17 2.34.45.9-.24.11V28a5 5 0 0 1-2.23 8.94l-
.02.06a8 8 0 0 1-7.75 6h-20a8 8 0 0 1-7.74-6l-.02-.06a5 5 0 0 1-2.24-8.94v-6.76l-.79-1.58-.44-
.9.9-.44.63-.32H240a23.01 23.01 0 0 1 44.37-2zm-36.82 2a1 1 0 0 0-.44.1l-3.1 1.56.89 1.79
1.31-.66a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9 0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .9
0l2.21-1.1a3 3 0 0 1 2.69 0l2.2 1.1a1 1 0 0 0 .86.02l2.88-1.27a3 3 0 0 1 2.43 0l2.88 1.27a1 1 0
0 0 .85-.02l3.1-1.55-.89-1.79-1.42.71a3 3 0 0 1-2.56.06l-2.77-1.23a1 1 0 0 0-.4-.09h-.01a1 1 0
0 0-.4.09l-2.78 1.23a3 3 0 0 1-2.56-.06l-2.3-1.15a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.1l-2.21
1.11a3 3 0 0 1-2.69 0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01a1 1 0 0 0-.44.1l-2.21 1.11a3 3 0 0 1-2.69
0l-2.2-1.1a1 1 0 0 0-.45-.11h-.01zm0-2h-4.9a21.01 21.01 0 0 1 39.61 0h-2.09l-.06-.13-.26.13h-
32.31zm30.35 7.68l1.36-.68h1.3v2h-36v-1.15l.34-.17 1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69
0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.69 0l1.36-.68h2.59l1.36.68a3 3 0 0 0 2.56.06l1.67-
.74h3.23l1.67.74a3 3 0 0 0 2.56-.06zM246.18 27l16.37 4.91L278.93 27h-32.75zm-.63
2h.34l16.66 5 16.67-5h.33a3 3 0 1 1 0 6h-34a3 3 0 1 1 0-6zm1.35 8a6 6 0 0 0 5.65 4h20a6 6 0 0
0 5.66-4H246.9z'/%3E%3Cpath d='M159.5 21.02A9 9 0 0 0 151 15h-42a9 9 0 0 0-8.5 6.02 6 6
0 0 0 .02 11.96A8.99 8.99 0 0 0 109 45h42a9 9 0 0 0 8.48-12.02 6 6 0 0 0 .02-11.96zM151 17h-
42a7 7 0 0 0-6.33 4h54.66a7 7 0 0 0-6.33-4zm-9.34 26a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7
7h-4.34a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7 7h-4.34a8.98 8.98 0 0 0 3.34-7h-2a7 7 0 0 1-7
7h-7a7 7 0 1 1 0-14h42a7 7 0 1 1 0 14h-9.34zM109 27a9 9 0 0 0-7.48 4H101a4 4 0 1 1 0-
8h58a4 4 0 0 1 0 8h-.52a9 9 0 0 0-7.48-4h-42z'/%3E%3Cpath d='M39 115a8 8 0 1 0 0-16 8 8 0
0 0 0 16zm6-8a6 6 0 1 1-12 0 6 6 0 0 1 12 0zm-3-29v-2h8v-6H40a4 4 0 0 0-4 4v10H22l-1.33
4-.67 2h2.19L26 130h26l3.81-40H58l-.67-2L56 84H42v-6zm-4-4v10h2V74h8v-2h-8a2 2 0 0
0-2 2zm2 12h14.56l.67 2H22.77l.67-2H40zm13.8 4H24.2l3.62 38h22.36l3.62-
38z'/%3E%3Cpath d='M129 92h-6v4h-6v4h-6v14h-3l.24 2 3.76 32h36l3.76-32 .24-2h-3v-14h-
6v-4h-6v-4h-8zm18 22v-12h-4v4h3v8h1zm-3 0v-6h-4v6h4zm-6 6v-16h-4v19.17c1.6-.7 2.97-
1.8 4-3.17zm-6 3.8V100h-4v23.8a10.04 10.04 0 0 0 4 0zm-6-.63V104h-4v16a10.04 10.04 0 0 0
4 3.17zm-6-9.17v-6h-4v6h4zm-6 0v-8h3v-4h-4v12h1zm27-12v-4h-4v4h3v4h1v-4zm-6 0v-8h-
4v4h3v4h1zm-6-4v-4h-4v8h1v-4h3zm-6 4v-4h-4v8h1v-4h3zm7 24a12 12 0 0 0 11.83-
10h7.92l-3.53 30h-32.44l-3.53-30h7.92A12 12 0 0 0 130 126z'/%3E%3Cpath d='M212 86v2h-
4v-2h4zm4 0h-2v2h2v-2zm-20 0v.1a5 5 0 0 0-.56 9.65l.06.25 1.12 4.48a2 2 0 0 0 1.94
1.52h.01l7.02 24.55a2 2 0 0 0 1.92 1.45h4.98a2 2 0 0 0 1.92-1.45l7.02-24.55a2 2 0 0 0 1.95-
1.52L224.5 96l.06-.25a5 5 0 0 0-.56-9.65V86a14 14 0 0 0-28 0zm4 0h6v2h-9a3 3 0 1 0 0
```

6H223a3 3 0 1 0 0-6H220v-2h2a12 12 0 1 0-24 0h2zm-1.44 14l-1-4h24.88l-1 4h-22.88zm8.95
26l-6.86-24h18.7l-6.86 24h-4.98zM150 242a22 22 0 1 0 0-44 22 22 0 0 0 0 44zm24-22a24 24 0
1 1-48 0 24 24 0 0 1 48 0zm-28.38 17.73l2.04-.87a6 6 0 0 1 4.68 0l2.04.87a2 2 0 0 0 2.5-
.82l1.14-1.9a6 6 0 0 1 3.79-2.75l2.15-.5a2 2 0 0 0 1.54-2.12l-.19-2.2a6 6 0 0 1 1.45-4.46l1.45-
1.67a2 2 0 0 0 0-2.62l-1.45-1.67a6 6 0 0 1-1.45-4.46l.2-2.2a2 2 0 0 0-1.55-2.13l-2.15-.5a6 6 0 0
1-3.8-2.75l-1.13-1.9a2 2 0 0 0-2.5-.8l-2.04.86a6 6 0 0 1-4.68 0l-2.04-.87a2 2 0 0 0-2.5.82l-1.14
1.9a6 6 0 0 1-3.79 2.75l-2.15.5a2 2 0 0 0-1.54 2.12l.19 2.2a6 6 0 0 1-1.45 4.46l-1.45 1.67a2 2 0
0 0 0 2.62l1.45 1.67a6 6 0 0 1 1.45 4.46l-.2 2.2a2 2 0 0 0 1.55 2.13l2.15.5a6 6 0 0 1 3.8
2.75l1.13 1.9a2 2 0 0 0 2.5.8zm2.82.97a4 4 0 0 1 3.12 0l2.04.87a4 4 0 0 0 4.99-1.62l1.14-1.9a4
4 0 0 1 2.53-1.84l2.15-.5a4 4 0 0 0 3.09-4.24l-.2-2.2a4 4 0 0 1 .97-2.98l1.45-1.67a4 4 0 0 0 0-
5.24l-1.45-1.67a4 4 0 0 1-.97-2.97l.2-2.2a4 4 0 0 0-3.09-4.25l-2.15-.5a4 4 0 0 1-2.53-1.84l-
1.14-1.9a4 4 0 0 0-5-1.62l-2.03.87a4 4 0 0 1-3.12 0l-2.04-.87a4 4 0 0 0-4.99 1.62l-1.14 1.9a4 4
0 0 1-2.53 1.84l-2.15.5a4 4 0 0 0-3.09 4.24l.2 2.2a4 4 0 0 1-.97 2.98l-1.45 1.67a4 4 0 0 0 0
5.24l1.45 1.67a4 4 0 0 1 .97 2.97l-.2 2.2a4 4 0 0 0 3.09 4.25l2.15.5a4 4 0 0 1 2.53 1.84l1.14
1.9a4 4 0 0 0 5 1.62l2.03-.87zM152 207a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm6 2a1 1 0 1 1 2 0 1 1 0 0
1-2 0zm-11 1a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-6 0a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm3-5a1 1 0 1 1 2 0 1
1 0 0 1-2 0zm-8 8a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm3 6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm0 6a1 1 0 1 1 2
0 1 1 0 0 1-2 0zm4 7a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm5-2a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm5 4a1 1 0 1
1 2 0 1 1 0 0 1-2 0zm4-6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm6-4a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-4-3a1
1 0 1 1 2 0 1 1 0 0 1-2 0zm4-3a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-5-4a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm-
24 6a1 1 0 1 1 2 0 1 1 0 0 1-2 0zm16 5a5 5 0 1 0 0-10 5 5 0 0 0 0 10zm7-5a7 7 0 1 1-14 0 7 7 0
0 1 14 0zm86-29a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm19 9a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1
0 0 1-1-1zm-14 5a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm-25 1a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-
2zm5 4a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm9 0a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-
1zm15 1a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm12-2a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-
2zm-11-14a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-19 0a1 1 0 0 0 0 2h2a1 1 0 0 0 0-
2h-2zm6 5a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-25 15c0-.47.01-.94.03-1.4a5 5 0 0
1-1.7-8 3.99 3.99 0 0 1 1.88-5.18 5 5 0 0 1 3.4-6.22 3 3 0 0 1 1.46-1.05 5 5 0 0 1 7.76-
3.27A30.86 30.86 0 0 1 246 184c6.79 0 13.06 2.18 18.17 5.88a5 5 0 0 1 7.76 3.27 3 3 0 0 1
1.47 1.05 5 5 0 0 1 3.4 6.22 4 4 0 0 1 1.87 5.18 4.98 4.98 0 0 1-1.7 8c.02.46.03.93.03 1.4v1h-
62v-1zm.83-7.17a30.9 30.9 0 0 0-.62 3.57 3 3 0 0 1-.61-4.2c.37.28.78.49 1.23.63zm1.49-4.61c-
.36.87-.68 1.76-.96 2.68a2 2 0 0 1-.21-3.71c.33.4.73.75 1.17 1.03zm2.32-4.54c-.54.86-1.03
1.76-1.49 2.68a3 3 0 0 1-.07-4.67 3 3 0 0 0 1.56 1.99zm1.14-1.7c.35-.5.72-.98 1.1-1.46a1 1 0 1
0-1.1 1.45zm5.34-5.77c-1.03.86-2 1.79-2.9 2.77a3 3 0 0 0-1.11-.77 3 3 0 0 1 4-2zm42.66 2.77c-
.9-.98-1.87-1.9-2.9-2.77a3 3 0 0 1 4.01 2 3 3 0 0 0-1.1.77zm1.34 1.54c.38.48.75.96 1.1 1.45a1
1 0 1 0-1.1-1.45zm3.73 5.84c-.46-.92-.95-1.82-1.5-2.68a3 3 0 0 0 1.57-1.99 3 3 0 0 1-.07
4.67zm1.8 4.53c-.29-.9-.6-1.8-.97-2.67.44-.28.84-.63 1.17-1.03a2 2 0 0 1-.2 3.7zm1.14 5.51c-
.14-1.21-.35-2.4-.62-3.57.45-.14.86-.35 1.23-.63a2.99 2.99 0 0 1-.6 4.2zM275 214a29 29 0 0 0-
57.97 0h57.96zM72.33 198.12c-.21-.32-.34-.7-.34-1.12v-12h-2v12a4.01 4.01 0 0 0 7.09

```
2.54c.57-.69.91-1.57.91-2.54v-12h-2v12a1.99 1.99 0 0 1-2 2 2 2 0 0 1-1.66-.88zM75 176c.38 0
.74-.04 1.1-.12a4 4 0 0 0 6.19 2.4A13.94 13.94 0 0 1 84 185v24a6 6 0 0 1-6 6h-3v9a5 5 0 1 1-
10 0v-9h-3a6 6 0 0 1-6-6v-24a14 14 0 0 1 14-14 5 5 0 0 0 5 5zm-17 15v12a1.99 1.99 0 0 0 1.22
1.84 2 2 0 0 0 2.44-.72c.21-.32.34-.7.34-1.12v-12h2v12a3.98 3.98 0 0 1-5.35 3.77 3.98 3.98 0 0
1-.65-.3V209a4 4 0 0 0 4 4h16a4 4 0 0 0 4-4v-24c.01-1.53-.23-2.88-.72-4.17-.43.1-.87.16-
1.28.17a6 6 0 0 1-5.2-3 7 7 0 0 1-6.47-4.88A12 12 0 0 0 58 185v6zm9 24v9a3 3 0 1 0 6 0v-9h-
6z'/%3E%3Cpath d='M-17 191a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm19 9a1 1 0 0 1 1-1h2a1 1 0 0
1 0 2H3a1 1 0 0 1-1-1zm-14 5a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm-25 1a1 1 0 0 0 0 2h2a1 1 0 0
0 0-2h-2zm5 4a1 1 0 0 0 0 2h2a1 1 0 0 0 0-2h-2zm9 0a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0
1-1-1zm15 1a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm12-2a1 1 0 0 0 0 2h2a1 1 0 0 0
0-2H4zm-11-14a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-19 0a1 1 0 0 0 0 2h2a1 1 0 0
0 0-2h-2zm6 5a1 1 0 0 1 1-1h2a1 1 0 0 1 0 2h-2a1 1 0 0 1-1-1zm-25 15c0-.47.01-.94.03-1.4a5 5
0 0 1-1.7-8 3.99 3.99 0 0 1 1.88-5.18 5 5 0 0 1 3.4-6.22 3 3 0 0 1 1.46-1.05 5 5 0 0 1 7.76-
3.27A30.86 30.86 0 0 1-14 184c6.79 0 13.06 2.18 18.17 5.88a5 5 0 0 1 7.76 3.27 3 3 0 0 1 1.47
1.05 5 5 0 0 1 3.4 6.22 4 4 0 0 1 1.87 5.18 4.98 4.98 0 0 1-1.7 8c.02.46.03.93.03 1.4v1h-62v-
1zm.83-7.17a30.9 30.9 0 0 0-.62 3.57 3 3 0 0 1-.61-4.2c.37.28.78.49 1.23.63zm1.49-4.61c-
.36.87-.68 1.76-.96 2.68a2 2 0 0 1-.21-3.71c.33.4.73.75 1.17 1.03zm2.32-4.54c-.54.86-1.03
1.76-1.49 2.68a3 3 0 0 1-.07-4.67 3 3 0 0 0 1.56 1.99zm1.14-1.7c.35-.5.72-.98 1.1-1.46a1 1 0 1
0-1.1 1.45zm5.34-5.77c-1.03.86-2 1.79-2.9 2.77a3 3 0 0 0-1.11-.77 3 3 0 0 1 4-2zm42.66 2.77c-
.9-.98-1.87-1.9-2.9-2.77a3 3 0 0 1 4.01 2 3 3 0 0 0-1.1.77zm1.34 1.54c.38.48.75.96 1.1 1.45a1
1 0 1 0-1.1-1.45zm3.73 5.84c-.46-.92-.95-1.82-1.5-2.68a3 3 0 0 0 1.57-1.99 3 3 0 0 1-.07
4.67zm1.8 4.53c-.29-.9-.6-1.8-.97-2.67.44-.28.84-.63 1.17-1.03a2 2 0 0 1-.2 3.7zm1.14 5.51c-
.14-1.21-.35-2.4-.62-3.57.45-.14.86-.35 1.23-.63a2.99 2.99 0 0 1-.6 4.2zM15 214a29 29 0 0 0-
57.97 0h57.96z'/%3E%3C/g%3E%3C/g%3E%3C/svg%3E");
}
```

**templates/index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Chatbot</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<link rel="stylesheet" href="{{ url_for('static', filename='styles/style.css') }}">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>
<body>
```

```
<!-- partial:index.partial.html -->
<section class="msger">
<header class="msger-header">
<div class="msger-header-title">
<i class="fas fa-bug"></i> Chatbot <i class="fas fa-bug"></i>
</div>
</header>
<main class="msger-chat">
<div class="msg left-msg">
<div class="msg-img" style="background-image:
url(https://image.flaticon.com/icons/svg/327/327779.svg)"></div>
<div class="msg-bubble">
<div class="msg-info">
<div class="msg-info-name">Chatbot</div>
<div class="msg-info-time">12:45</div>
</div>
<div class="msg-text">

Hi, welcome to ChatBot! Go ahead and send me a message.
</div>
</div>
</div>
</main>
<form class="msger-inputarea">
<input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">
<button type="submit" class="msger-send-btn">Send</button>
</form>
</section>
<!-- partial -->
<script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>
<script>
const msgerForm = get(".msger-inputarea");
const msgerInput = get(".msger-input");
const msgerChat = get(".msger-chat");
// Icons made by Freepik from www.flaticon.com
const BOT_IMG = "https://image.flaticon.com/icons/svg/327/327779.svg";
const PERSON_IMG = "https://image.flaticon.com/icons/svg/145/145867.svg";
const BOT_NAME = " ChatBot";
const PERSON_NAME = "You";
msgerForm.addEventListener("submit", event => {
```

```
event.preventDefault();
const msgText = msgerInput.value;
if (!msgText) return;
appendMessage(PERSON_NAME, PERSON_IMG, "right", msgText);
msgerInput.value = "";
botResponse(msgText);
});
function appendMessage(name, img, side, text) {
// Simple solution for small apps
const msgHTML = `
<div class="msg ${side}-msg">
<div class="msg-img" style="background-image: url(${img})"></div>
<div class="msg-bubble">
<div class="msg-info">
<div class="msg-info-name">${name}</div>
<div class="msg-info-time">${formatDate(new Date())}</div>
</div>
<div class="msg-text">${text}</div>
</div>
</div>
`;
msgerChat.insertAdjacentHTML("beforeend", msgHTML);
msgerChat.scrollTop += 500;
}
function botResponse(rawText) {
// Bot Response
$.get("/get", { msg: rawText }).done(function (data) {
console.log(rawText);
console.log(data);
const msgText = data;
appendMessage(BOT_NAME, BOT_IMG, "left", msgText);
});
}
// Utils
function get(selector, root = document) {
return root.querySelector(selector);
}
function formatDate(date) {
const h = "0" + date.getHours();
const m = "0" + date.getMinutes();
return `${h.slice(-2)}:${m.slice(-2)}`;
}
</script>
</body>
</html>
```

**app.py**

```
import nltk
nltk.download('popular')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('model.h5')
import json
import random
intents = json.loads(open('data.json').read())
words = pickle.load(open('texts.pkl','rb'))
classes = pickle.load(open('labels.pkl','rb'))
def clean_up_sentence(sentence):
# tokenize the pattern - split words into array
sentence_words = nltk.word_tokenize(sentence)
# stem each word - create short form for word
sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):
# tokenize the pattern
sentence_words = clean_up_sentence(sentence)
# bag of words - matrix of N words, vocabulary matrix
bag = [0]*len(words)
for s in sentence_words:
for i,w in enumerate(words):
if w == s:
# assign 1 if current word is in the vocabulary position
bag[i] = 1
if show_details:
```

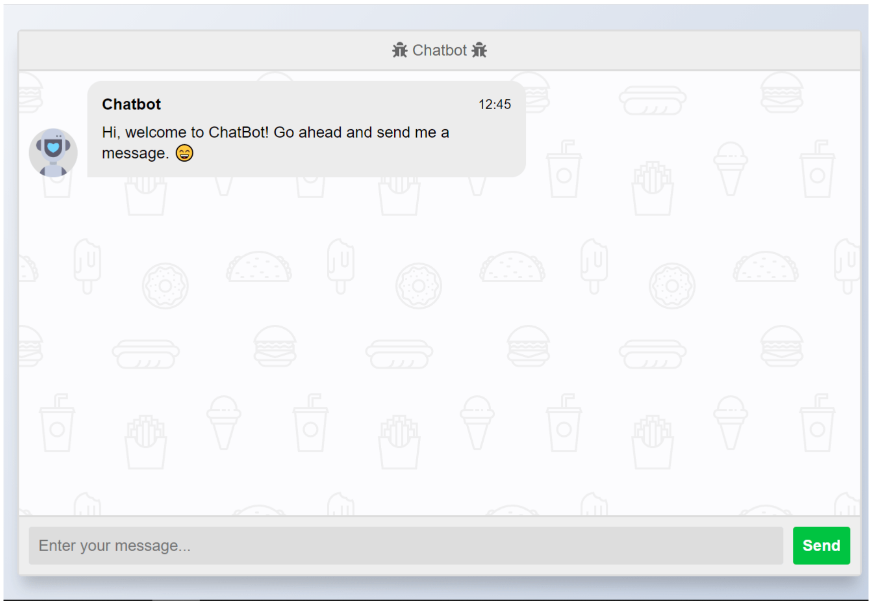
```
print ("found in bag: %s" % w)
return(np.array(bag))
def predict_class(sentence, model):
# filter out predictions below a threshold
p = bow(sentence, words,show_details=False)
res = model.predict(np.array([p]))[0]
ERROR_THRESHOLD = 0.25
results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
# sort by strength of probability
results.sort(key=lambda x: x[1], reverse=True)
return_list = []
for r in results:
return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
return return_list
def getResponse(ints, intents_json):
tag = ints[0]['intent']
list_of_intents = intents_json['intents']
for i in list_of_intents:
if(i['tag']== tag):
result = random.choice(i['responses'])
break
return result
def chatbot_response(msg):
ints = predict_class(msg, model)
res = getResponse(ints, intents)
return res
from flask import Flask, render_template, request
app = Flask(__name__)
app.static_folder = 'static'
@app.route("/")
def home():
return render_template("index.html")
@app.route("/get")
def get_bot_response():
userText = request.args.get('msg')
return chatbot_response(userText)
if __name__ == "__main__":
app.run()
```

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve a random response from the list of responses.

Again we import the necessary packages and load the ‘texts.pkl’ and ‘labels.pkl’ pickle files which we have created when we trained our model:

To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class. After predicting the class, we will get a random response from the list of intents.

OUTPUT:



Chatbot 12:45

Hi, welcome to ChatBot! Go ahead and send me a message. 😄