Course : Artificial Intelligence

Project : Create a Chatbot in Python

Topic: Start create a chatbot in python model by loading and pre-processing the dataset.

Introduction :

Chatbot is a computer program that humans will interact with in natural spoken language and including artificial intelligence techniques such as NLP (Natural language processing) that makes the chatbot more interactive and more reliable. Based on the recent epidemiological situation, the increasing demand and reliance on electronic education has become very difficult to access to the university due to the curfew imposed, and this has led to limited access to information for academics at the university.

This project aims to build a chatbot for Admission and Registration to answer every person who asks about the university, colleges, majors and admission policy.

Import Libraries:

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

1. **Data Loading**: First, you need to load your dataset. This could be from a CSV file, a database, or any other source. In Python, you might use libraries like `pandas` for this task.

pythonCopy code

```
import pandas as pd
dataset = pd.read_csv('your_dataset.csv'
```

2. **Data Exploration**: After loading the data, it's a good practice to explore it. Check for the number of rows and columns, data types, and basic statistics.

pythonCopy code

```
print(dataset.shape)
print dataset.info())
print dataset.describe
```

3. **Handling Missing Data**: You may need to handle missing data by filling in missing values or removing rows with missing values.

pythonCopy code

```
dataset.dropna() # To remove rows with missing values
 dataset.fillna(value) # To fill in missing values
```

4. **Data Preprocessing**: This step involves transforming the data as needed. Common preprocessing steps include encoding categorical variables, scaling numerical features, and feature engineering.

pythonCopy code

```
from sklearn.preprocessing import
 LabelEncoder, StandardScaler label_encoder = LabelEncoder()
 dataset['categorical_column'] = label_encoder.fit_transform(dataset['categorical_column'])
scaler = StandardScaler() dataset['numeric_column'] = scaler.fit_transform(dataset 'numeric_column'].values.reshape(-1,
1))
```

5. **Data Splitting**: If you're working on a supervised learning task, split the dataset into training and testing sets.

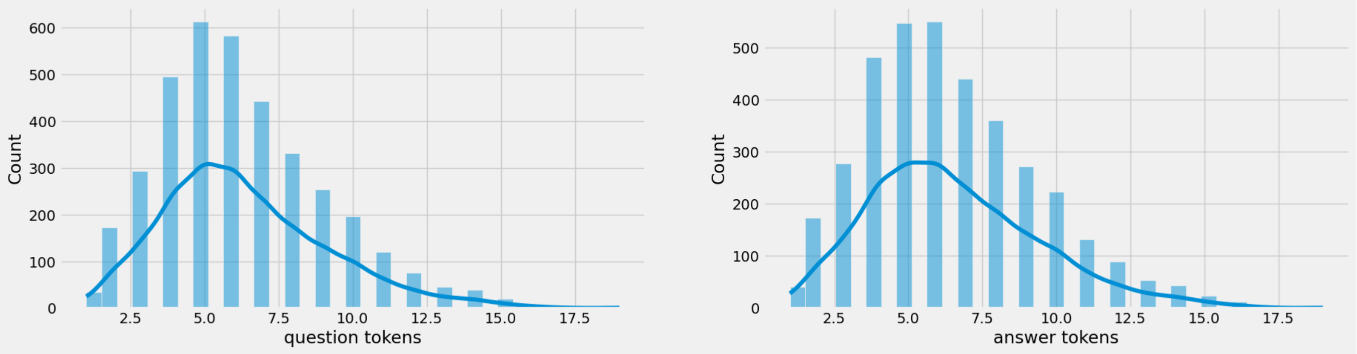
pythonCopy code

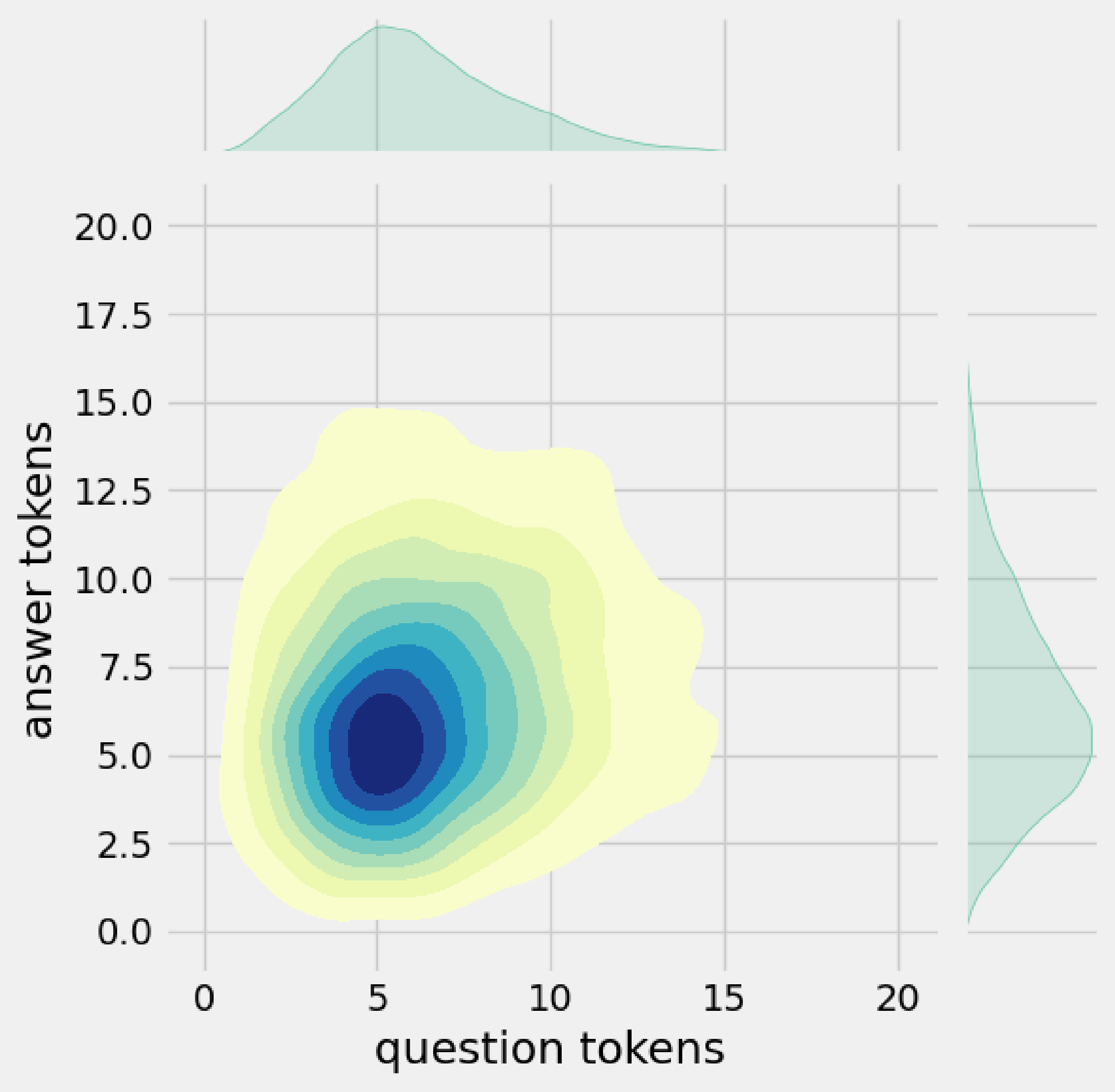
```
from sklearn.model_selection import train_test_split X = dataset.drop('target_column', axis=1) y = dataset 'target_column']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. **Feature Selection**: Choose relevant features for your model if necessary.
7. **Data Augmentation** (for certain tasks like image processing).
8. **Data Normalization**: Normalize the data if needed.
9. **Data Loading and Augmentation**: If you're working with deep learning, you might use libraries like TensorFlow or PyTorch to load and augment your data.

Data Visualization

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





Tokenization

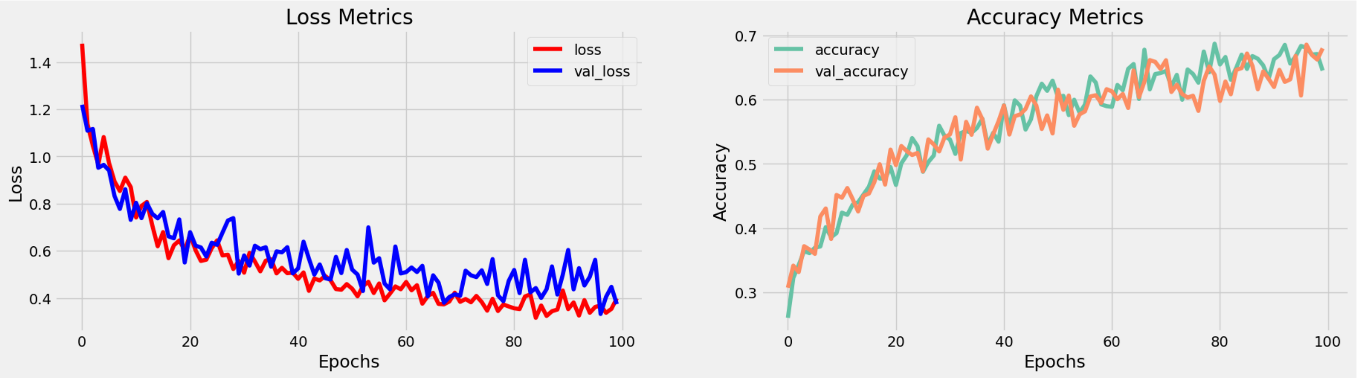
```
vectorize_layer=TextVectorization(
    max_tokens=vocab_size,
    standardize=None,
    output_mode='int',
    output_sequence_length=max_sequence_length
)
vectorize_layer.adapt(df['encoder_inputs']+' '+df['decoder_targets']+' <start> <end>')
vocab_size=len(vectorize_layer.get_vocabulary())
print(f'Vocab size: {len(vectorize_layer.get_vocabulary())}')
print(f'{vectorize_layer.get_vocabulary()[:12]}')
```

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

Visualize Metrics

```
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

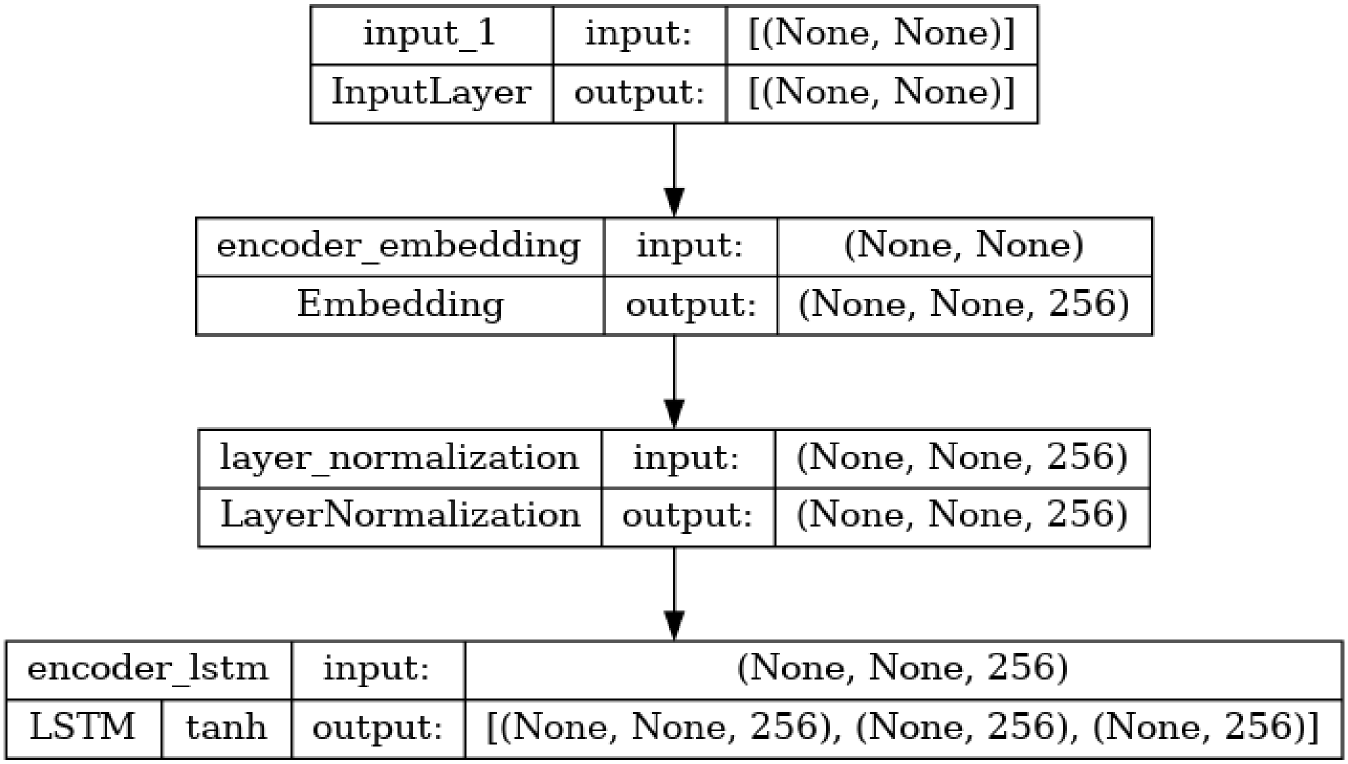
```
model.load_weights('ckpt')
model.save('models',save_format='tf')

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.layers.core.embedding.Embedding object at 0x782084b9d190>
<keras.layers.normalization.layer_normalization.LayerNormalization object at 0x7820e56f1
b90>
<keras.layers.rnn.lstm.LSTM object at 0x7820841bd650>
---------------------
Decoder layers:
<keras.layers.core.embedding.Embedding object at 0x78207c258590>
<keras.layers.normalization.layer_normalization.LayerNormalization object at 0x78207c78b
d10>
<keras.layers.rnn.lstm.LSTM object at 0x78207c258a10>
<keras.layers.core.dense.Dense object at 0x78207c2636d0>
---------------------
```

Create Inference Model

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

Out[20]:



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)]

Time to Chat

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

Conclusion

This bot was built to respond to the inquiries of the Tawjihi students regarding each of the university's faculties and their specializations, with extracted information for each specialization, familiarizing students with the level exams that students submit about their enrollment in the university, introducing the educational qualification diploma program and the mechanism for joining it. Giving students notes on the electronic enrollment application package, the locations of approved banks, and how to fill out the application.