CREATE CHATBOT IN PYTHON

Phase 3 : Development Part 1

INTRODUCTION:

A chatbot is a computer program that simulates human conversation. It is a type of artificial intelligence (AI) that can be used to create virtual assistants, customer service representatives, and other interactive applications.

- Chatbots typically use natural language processing (NLP) to understand and respond to user input. This means that they can be trained to recognize and respond to a wide range of different human languages and dialects.
- AI-powered chatbots use a variety of AI techniques, including natural language processing (NLP), machine learning, and deep learning.
- NLP allows chatbots to understand the meaning of text, machine learning allows chatbots to learn from their experiences, and deep learning allows chatbots to generate more natural and engaging responses

Development Phase:

In this Phase begin building your project by loading and preprocessing the dataset. Start building the chatbot by preparing the environment and implementing basic user interactions.Install required libraries, like transformers for GPT-3 integration and flask for web app development.

DATA SOURCE:

The provided dataset consists of multiple short conversations between two participants, primarily focusing on casual and friendly exchanges. The conversations cover topics like greetings, well-being, attending school at PCC (a specific institution), and some small talk about the weather. Each conversation is structured as a back-and-forth dialogue between the participants, featuring natural language and informal communication

Dataset Link :(https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot)

The dataset you have provided is an example of a conversational dataset. Conversational datasets are typically used to train chatbot models. The dataset contains a dialogue between two people, where each person takes a turn speaking. The dataset also includes the context of the conversation, which is important for training chatbot models to generate natural and informative responses.

Example of Dataset:

```
hi, how are you doing?	i'm fine. how about yourself?
i'm fine. how about yourself?	i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.	no problem. so how have you been?
no problem. so how have you been?	i've been great. what about you?
i've been great. what about you?	i've been good. i'm in school right now.
i've been good. i'm in school right now.	what school do you go to?
what school do you go to?	i go to pcc.
i go to pcc.	do you like it there?
do you like it there?	it's okay. it's a really big campus.
it's okay. it's a really big campus.	good luck with school.
good luck with school.	thank you very much.
how's it going?	i'm doing well. how about you?
i'm doing well. how about you?	never better, thanks.
never better, thanks.	so how have you been lately?
so how have you been lately?	i've actually been pretty good. you?
i've actually been pretty good. you?	i'm actually in school right now.
i'm actually in school right now.	which school do you attend?
which school do you attend?	i'm attending pcc right now.
i'm attending pcc right now.	are you enjoying it there?
are you enjoying it there?	it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.	good luck with that.
good luck with that.	thanks.
how are you doing today?	i'm doing great. what about you?
i'm doing great. what about you?	i'm absolutely lovely, thank you.
i'm absolutely lovely, thank you.	everything's been good with you?
everything's been good with you?	i haven't been better. how about yourself?
i haven't been better. how about yourself?	i started school recently.
```

Create python Environment:

Create a Python virtual environment (venv) using the following command in your terminal

```
Python -m venv myenv
```

This allows you to isolate your project’s dependencies and packages, making it easier to manage and maintain different Python projects with their own specific requirements.

Install the Package:

We going to pip package manager to install package.and it stands for “Pip Installs Packages.” It is a tool that simplifies the process of installing, managing, and uninstalling Python packages and libraries.

Package:

NumPy, which stands for “Numerical Python,” is a fundamental Python library for numerical and scientific computing. It provides support for working with large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently

```
pip install numpy
```

Pandas is a popular open-source data manipulation and analysis library for Python. It provides easy-to-use data structures and data analysis tools that make working with structured data, such as tables and spreadsheets, efficient and convenient

```
pip install pandas as pd
```

Matplotlib is a popular Python library used for creating static, animated, or interactive visualizations in various formats

```
pip install matplotlib
```

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab (FAIR). It has gained widespread popularity in the field of machine learning and artificial intelligence due to its flexibility, dynamic computation graph, and strong support for neural networks.

pip install pytorch

Hugging Face Transformers is a popular open-source library and platform that provides a wide range of pre-trained natural language processing (NLP) models, particularly transformer-based models.

pip install transformers

Flask is a lightweight and versatile Python web framework that is widely used for developing web applications

pip install flask

Importing the Package:

We can use installed packages,by importing the packages,

```
import pytorch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re,string
```

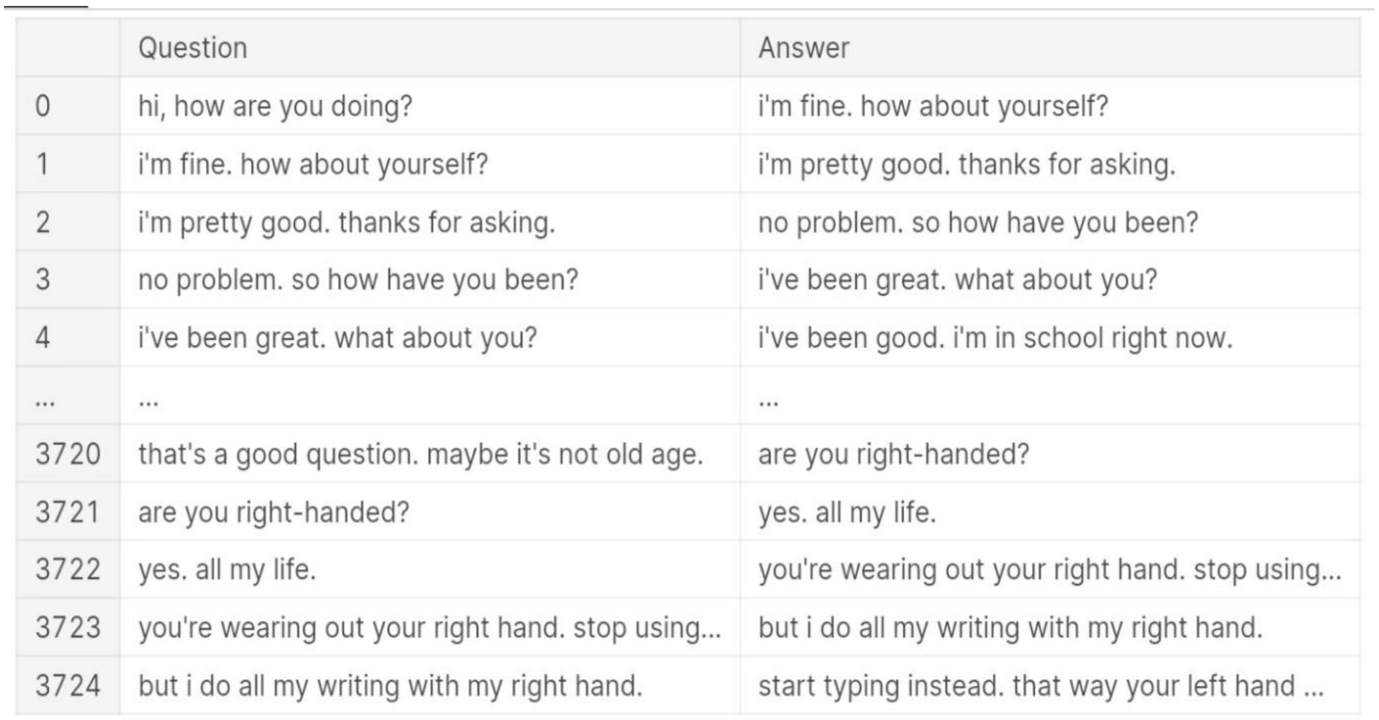
From transformers we are importing the GPT2Tokenizer and GPT2LMHeadMODEL.The GPT2Tokenizer is a component of the Hugging Face Transformers library, specifically designed for tokenizing text data for models like GPT-2.

The GPT2LMHeadModel is a specific model class within Hugging Face's Transformers library, and it's designed for natural language generation tasks using the GPT-2 architectures

Data Preprocessing :

We go read the file using the pandas dataframe the we get the dataset file from the kagge(link).

```
Df=pd.read_csv('dialogs.txt',sep='\t',names=['Question' , 'Answer'])
```



	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.
...	...	...
3720	that's a good question. maybe it's not old age.	are you right-handed?
3721	are you right-handed?	yes. all my life.
3722	yes. all my life.	you're wearing out your right hand. stop using...
3723	you're wearing out your right hand. stop using...	but i do all my writing with my right hand.
3724	but i do all my writing with my right hand.	start typing instead. that way your left hand ...

Clean the dataset by removing the number and replace with the whitespace by declaring the function, insert in dataframe

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\d+', ' ', text)  # Replace all digits with spaces
    text = re.sub(r'([^\w\s])', r' \1 ', text)  # Add a space before and a
fter each punctuation character
    text = re.sub(r'\s+', ' ', text)
    text = text.strip()
    return text


df['Encoder Inputs']=df['Question'].apply(clean_text)
df['Decoder Inputs']="<sos> " + df['Answer'].apply(clean_text) + ' <eos>'
df["Decoder Targets"] = df['Answer'].apply(clean_text) + ' <eos>'


df.head()
```

The Encoded input and Decoder input are use to generate the token in tokenizer and the Decoder Targets are use labels or Target for model for the Training.

Data Visualization:

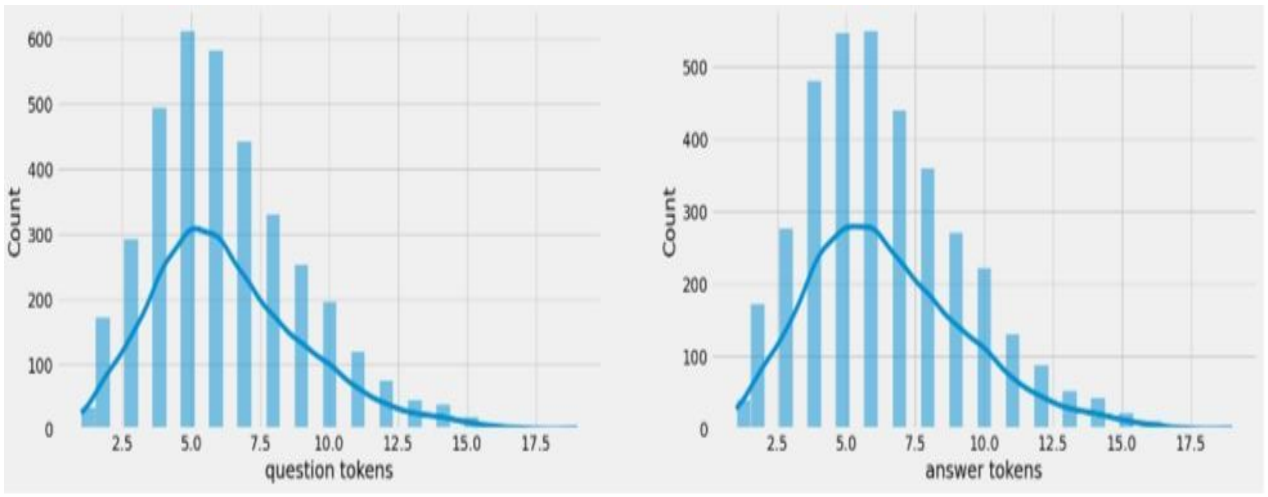
We are use the matplotlib and seaborn for the Data Visualization

Before data processing:

Import matplotlib.pyplot as plt

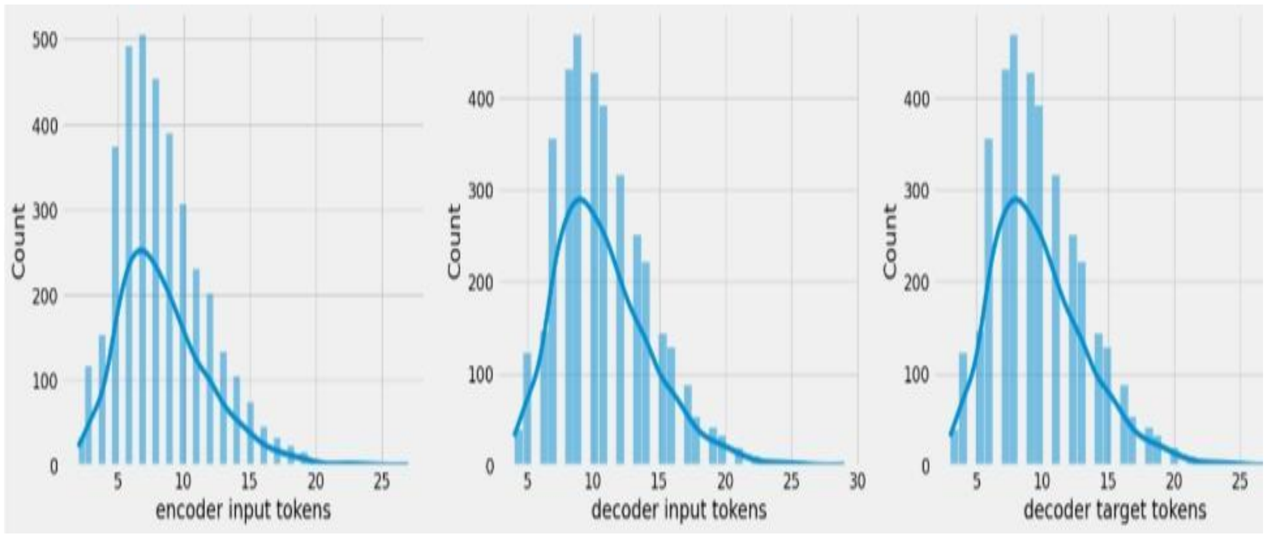
Import seaborn as sns

```
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',fil
l=True,cmap='YlGnBu')

plt.show()
```



After data Processing :

```
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.spl
it()))
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,k
ind='kde',fill=True,cmap='YlGnBu')
plt.show()
```



Conclusion:

We create dataset for model training .we have use the numpy, pandas for read file and string for cleaning the text   Matplotlib and seabron for data visualization