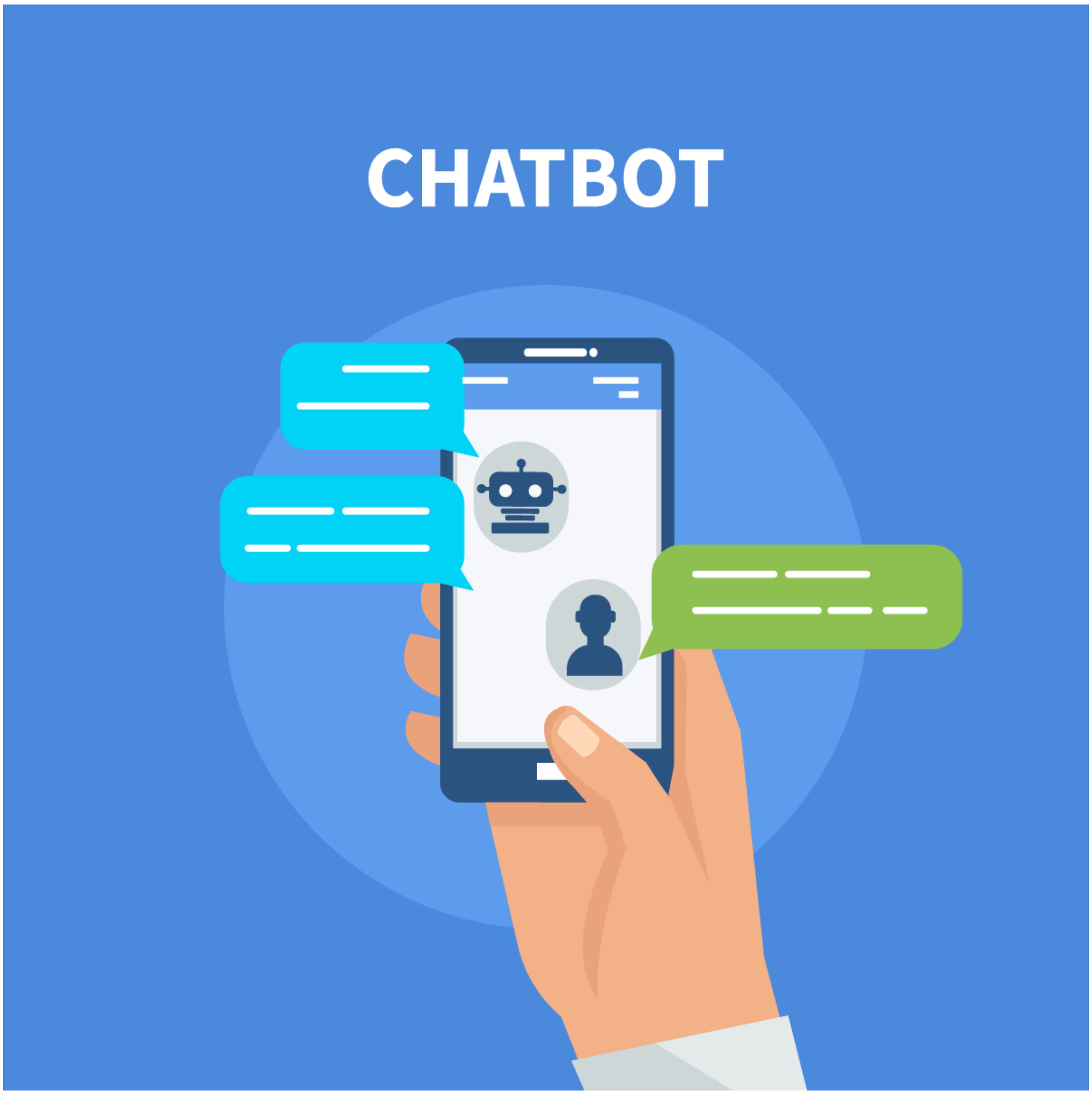
CREATE A CHATBOT IN PYTHON



Presented by:-

Ahila. K

Bhuvaneshwari. S

Gopika. K

Harshini. A

Kavya. K

Table Of Contents:

Introduction:

A chatbot is a computer program designed to simulate human conversation. In Python, chatbots are typically created using natural language processing (NLP) and machine learning techniques. They can be integrated into various applications, websites, or messaging platforms to interact with users in a conversational manner.

We will create an AI chatbot using Natural Language Processing (NLP) in Python. Our goal is to help you build a smart chatbot. First, we'll explain NLP, which helps computers understand human language. Then, we'll show you how to use AI to make a chatbot to have real conversations with people.

Choose a Framework or Library:

There are various libraries and frameworks like NLTK, spaCy, and TensorFlow for NLP and machine learning. You can choose one that suits your project's requirements.

Data Preparation:

Collect and preprocess data for training your chatbot. This can include a dataset of questions and answers or dialogues.

NLP and ML Models:

Train or fine-tune NLP models like RNNs, LSTMs, or transformers to understand and generate text. You may use pre-trained models for this purpose.

User Interface:

Create a user interface to interact with the chatbot, whether it's a command-line interface or a chat widget on a website.

Conversational Logic:

Implement the logic that handles user input and generates relevant responses. This can include rulebased systems or more sophisticated machine learning models.

Testing and Deployment:

Test your chatbot extensively and deploy it to your desired platform or application.

Training AI Model For Chatbot Using Python:

Generating insights:

We will create an AI chatbot using Natural Language Processing (NLP) in Python. Our goal is to help you build a smart chatbot. First, we'll explain NLP, which helps computers understand human language. Then, we'll show you how to use AI to make a chatbot to have real conversations with people.

Installing Packages required to Build AI Chatbot:

We will begin by installing a few libraries which are as follows :

Code:

```
# To be able to convert text to Speech
! pip install SpeechRecognition  #(3.8.1)
#To convey the Speech to text and also speak it out
!pip install gTTS  #(2.2.3)
# To install our language model
!pip install transformers  #(4.11.3)
!pip install tensorflow #(2.6.0, or pytorch)
```

We will start by importing some basic functions:

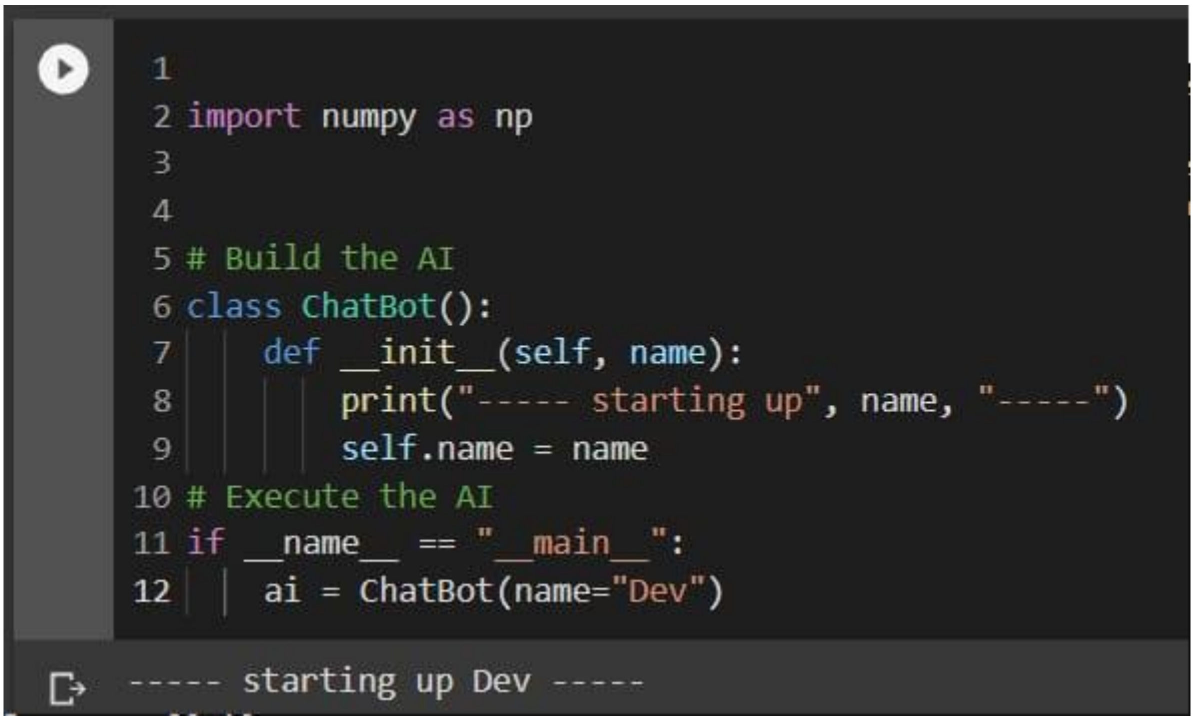
```
import numpy as np
```

We will begin by creating an empty class which we will build step by step. To build the chatbot, we would need to execute the full script. The name of the bot will be "Dev".

Code:

```
#Beginning of the AI
class ChatBot():
   def _init_(self, name):
        print("----- starting up", name, "-----")
        self.name = name
# Execute the AI
if _name_ == "_main_":
    ai = ChatBot(name="Dev")
```

Output:



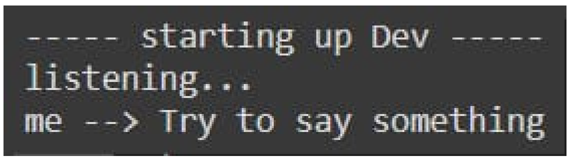
```

import numpy as np


# Build the AI
class ChatBot():
    def __init__(self, name):
        print("----- starting up", name, "-----")
        self.name = name
# Execute the AI
if __name__ == "__main__":
    ai = ChatBot(name="Dev")
```

```
----- starting up Dev -----
```

Speech to Text Conversion:

```
# Execute the AI
if _name_ == "_main_":
     ai = ChatBot(name="Dev")
     while True:
         ai.speech_to_text()
```

Output:



```
----- starting up Dev -----
listening...
me --> Try to say something
```

Fine-tuning Bot Responses:

After the chatbot hears its name, it will formulate a response accordingly and say something back. For this, the chatbot requires a text-to-speech module as well. Here, we will be using GTTS or Google Text to Speech library to save mp3 files on the file system which can be easily played back.

The following functionality needs to be added to our class so that the bot can respond back

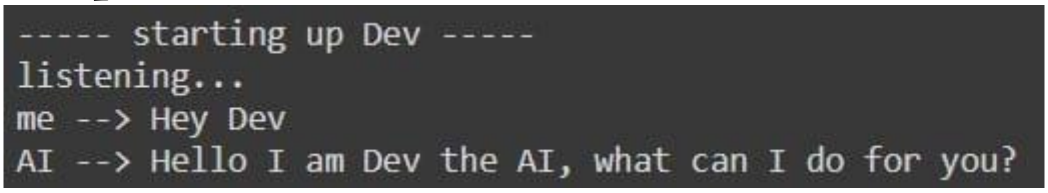
Code:

```
from gtts import gTTS

import os

@staticmethod

def text_to_speech(text):

   print("AI --> ", text)
   speaker = gTTS(text=text, lang="en", slow=False)

   speaker.save("res.mp3")

   os.system("start res.mp3")  #if you have a macbook->afplay
or for windows use->start

   os.remove("res.mp3")
```

Code:

```
#Those two functions can be used like this

# Execute the AI

if _name_ == "_main_":

    ai = ChatBot(name="Dev")

    while True:

       ai.speech_to_text()

       ## wake up

       if ai.wake_up(ai.text) is True:

          res = "Hello I am Dev the AI, what can I do for you?"

       ai.text_to_speech(res)
```

Output :



```
----- starting up Dev -----
listening...
me --> Hey Dev
AI --> Hello I am Dev the AI, what can I do for you?
```

Next, we can consider upgrading our chatbot to do simple commands like some of the virtual assistants help you to do.

An example of such a task would be to equip the chatbot to be able to answer correctly whenever the user asks for the current time.

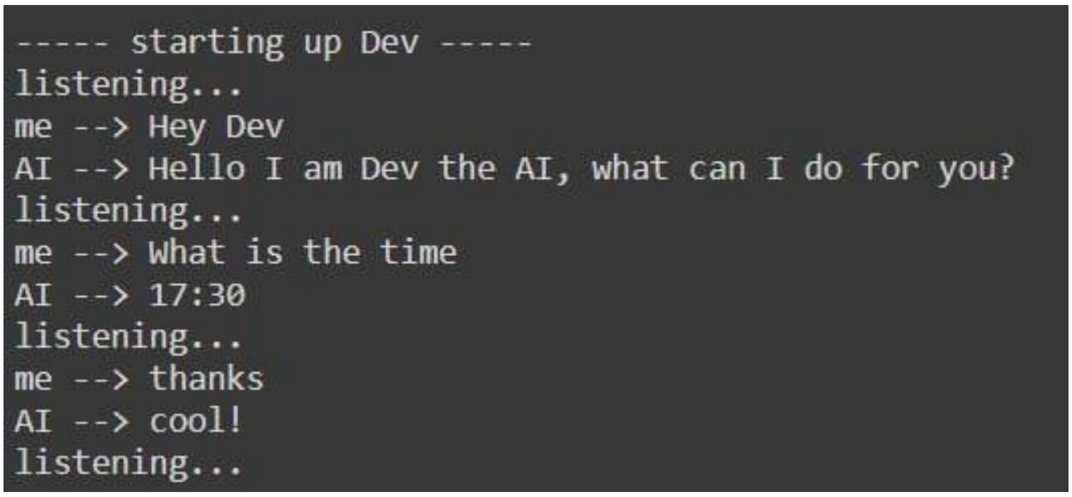
To add this function to the chatbot class, follow along with the code given below:

Code:

```
import datetime
@staticmethod
def action_time():
   return datetime.datetime.now().time().strftime('%H:%M')
# Run the AI
if _name_ == "_main_":
ai = ChatBot(name="Dev")
while True:
        ai.speech_to_text()
        ## waking up
        if ai.wake_up(ai.text) is True:
```

```
        res = "Hello I am Dev the AI, what can I do for you?"
      ## do any action
      elif "time" in ai.text:
        res = ai.action_time()
      ## respond politely
      elif any(i in ai.text for i in ["thank","thanks"]):
        res = np.random.choice(
            ["you're welcome!","anytime!",
             "no problem!","cool!",
             "I'm here if you need me!","peace out!"])
      ai.text_to_speech(res)
```

Output:



```
----- starting up Dev -----
listening...
me --> Hey Dev
AI --> Hello I am Dev the AI, what can I do for you?
listening...
me --> What is the time
AI --> 17:30
listening...
me --> thanks
AI --> cool!
listening...
```

After all of the functions that we have added to our chatbot, it can now use speech recognition techniques to respond to speech cues and reply with predetermined responses.

It is now time to incorporate artificial intelligence into our chatbot to create intelligent responses to human speech interactions with the chatbot or the ML model trained using NLP or Natural Language Processing.

The Language Model for AI Chatbot:

Here, we will use a Transformer Language Model for our chatbot. This model was presented by Google and it replaced the earlier traditional sequence to sequence models with attention mechanisms. This language model dynamically understands speech and its undertones. Hence, the model easily performs NLP tasks. Some of the most popularly used language models are Google's BERT and OpenAI's GPT. These models have multidisciplinary functionalities and billions of parameters which helps to improve the chatbot and make it truly intelligent.

The main package that we will be using in our code here is the Transformers package provided by Hugging Face. This tool is popular amongst developers as it provides tools that are pre-trained and ready to work with a variety of NLP tasks. In the code below, we have specifically used the Dialog GPT trained and created by Microsoft based on millions of conversations and ongoing chats on the Reddit platform in a given interval of time.

Code:

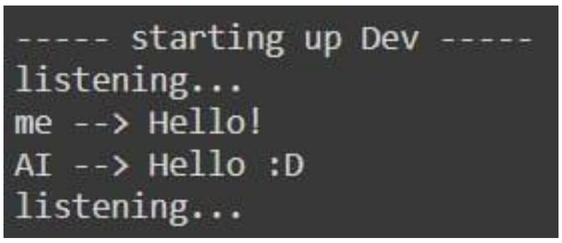
```
import transformers
nlp = transformers.pipeline("conversational",
#Time to try it out
input_text = "hello!"
nlp(transformers.Conversation(input_text),
pad_token_id=50256)
```

Hence you need to extract only the response of the chatbot here.

Code:

```
chat = nlp(transformers.Conversation(ai.text),
pad_token_id=50256)
res = str(chat)
res = res[res.find("bot >> ")+6:].strip()
```

Output:

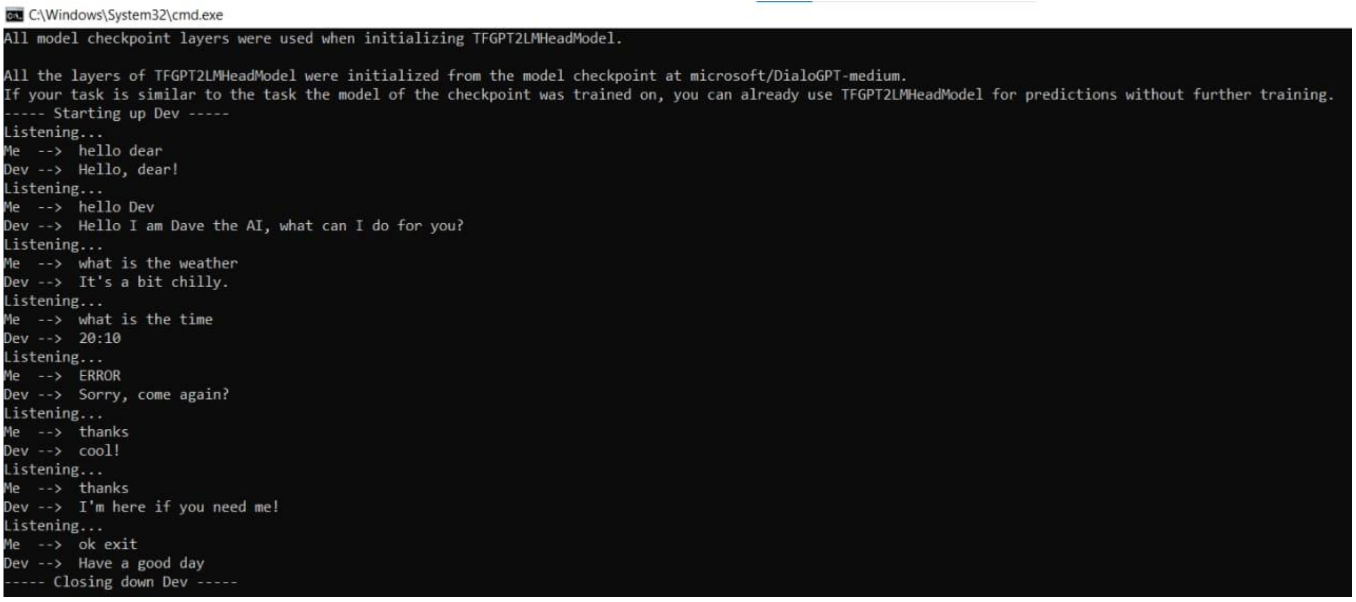


```
----- starting up Dev -----
listening...
me --> Hello!
AI --> Hello :D
listening...
```

Final code:

```
# for speech-to-text
import speech_recognition as sr
# for text-to-speech
from gtts import gTTS
# for language model
import transformers
import os
import time
# for data
import os
import datetime
import numpy as np
# Building the AI
class ChatBot():
    def _init_(self, name):
        print("----- Starting up", name, "-----")
        self.name = name
    def speech_to_text(self):
        recognizer = sr.Recognizer()
        with sr.Microphone() as mic:
            print("Listening...")
            audio = recognizer.listen(mic)
            self.text="ERROR"
        try:
            self.text = recognizer.recognize_google(audio)
            print("Me  --> ", self.text)
        except:
            print("Me  -->  ERROR")
    @staticmethod
    def text_to_speech(text):
        print("Dev --> ", text)
        speaker = gTTS(text=text, lang="en", slow=False)
        speaker.save("res.mp3")
        statbuf = os.stat("res.mp3")
        mbytes = statbuf.st_size / 1024
        duration = mbytes / 200
        os.system('start res.mp3')  #if you are using mac->afplay
or else for windows->start
        # os.system("close res.mp3")
        time.sleep(int(50*duration))
        os.remove("res.mp3")
    def wake_up(self, text):
        return True if self.name in text.lower() else False
    @staticmethod
    def action_time():
        return
datetime.datetime.now().time().strftime('%H:%M')
# Running the AI
if _name_ == "_main_":
    ai = ChatBot(name="dev")
    nlp = transformers.pipeline("conversational",
model="microsoft/DialoGPT-medium")
    os.environ["TOKENIZERS_PARALLELISM"] = "true"
    ex=True
    while ex:
        ai.speech_to_text()
        ## wake up
        if ai.wake_up(ai.text) is True:
            res = "Hello I am Dave the AI, what can I do for you?"
        ## action time
        elif "time" in ai.text:
            res = ai.action_time()
        ## respond politely
        elif any(i in ai.text for i in ["thank","thanks"]):
          res = np.random.choice(["you're
welcome!","anytime!","no problem!","cool!","I'm here if you
need me!","mention not"])
        elif any(i in ai.text for i in ["exit","close"]):
          res = np.random.choice(["Tata","Have a good
day","Bye","Goodbye","Hope to meet soon","peace out!"])
          ex=False
        ## conversation
        else:
          if ai.text=="ERROR":
            res="Sorry, come again?"
          else:
            chat = nlp(transformers.Conversation(ai.text),
pad_token_id=50256)
            res = str(chat)
            res = res[res.find("bot >> ")+6:].strip()
        ai.text_to_speech(res)
    print("----- Closing down Dev-----")
```

Final output:



```
C:\Windows\System32\cmd.exe
All model checkpoint layers were used when initializing TFGPT2LMHeadModel.

All the layers of TFGPT2LMHeadModel were initialized from the model checkpoint at microsoft/DialoGPT-medium.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFGPT2LMHeadModel for predictions without further training.
----- Starting up Dev -----
Listening...
Me  -->  hello dear
Dev -->  Hello, dear!
Listening...
Me  -->  hello Dev
Dev -->  Hello I am Dave the AI, what can I do for you?
Listening...
Me  -->  what is the weather
Dev -->  It's a bit chilly.
Listening...
Me  -->  what is the time
Dev -->  20:10
Listening...
Me  -->  ERROR
Dev -->  Sorry, come again?
Listening...
Me  -->  thanks
Dev -->  cool!
Listening...
Me  -->  thanks
Dev -->  I'm here if you need me!
Listening...
Me  -->  ok exit
Dev -->  Have a good day
----- Closing down Dev -----
```

Conclusion:

We’ve provided a step-by-step tutorial for creating a conversational chatbot. You can use this chatbot as a foundation for developing one that communicates like a human. The code samples we’ve shared are versatile and can serve as building blocks for similar chatbot projects.