**Problem Statement –** Develop an elementary chatbot for any suitable customer interaction application**.**

# for speech-to-text

import speech\_recognition as sr

# for text-to-speech

from gtts import gTTS

from playsound import playsound

# for language model

import transformers

import os

import time

# for data

import datetime

import numpy as np

# Building the AI

class ChatBot:

def \_\_init\_\_(self, channel):

print(f"----- Starting up {channel} -----")

self.channel = channel

def speech\_to\_text(self, only\_text=False):

if only\_text:

print("Me --> ", end="")

self.text = input()

return

recognizer = sr.Recognizer()

with sr.Microphone() as mic:

recognizer.adjust\_for\_ambient\_noise(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")

def text\_to\_speech(self, text, only\_text=False):

print(f"{self.channel} --> {text}")

if only\_text:

return

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

playsound("res.mp3")

time.sleep(int(50 \* duration))

os.remove("res.mp3")

def wake\_up(self, text):

return True if self.channel.lower() in text.lower() else False

@staticmethod

def action\_time():

return datetime.datetime.now().time().strftime("%H:%M")

# Running the AI

if \_\_name\_\_ == "\_\_main\_\_":

channel = "Dev"

ai = ChatBot(channel=channel)

nlp = transformers.pipeline("conversational", model="microsoft/DialoGPT-medium")

os.environ["TOKENIZERS\_PARALLELISM"] = "true"

ex = True

while ex:

ai.speech\_to\_text(only\_text=True)

## 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", "bye"]):

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()

try:

ai.text\_to\_speech(res, only\_text=False)

except:

pass

print(f"----- Closing down {channel} -----")

"""

OUTPUT :-

----- Starting up Dev -----

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.

Me --> Hey

Dev --> Hey! How are you?

Me --> I'm fine. what about you?

Dev --> I'm fine too.

Me --> what day is today?

Dev --> Today is the day.

Me --> nice joke

Dev --> I'm not sure if you're being serious or not.

Me --> what is current time?

Dev --> 20:43

Me --> who is the first president of india?

Dev --> I think it was Rajendra Prasad.

Me --> wrong

Dev --> I'm not sure what you mean.

Me --> who is the first prime minister of britain?

Dev --> Robert Walpole

Me --> bye

Dev --> Peace out!

----- Closing down Dev -----

"""