LOVELY PROFESSIONAL UNIVERSITY JALANDHAR(INDIA)



TOPIC - Personal Voice Assistant

Submitted By-

(1.)Vishal Gupta(A14/11814743)

(2.)Dipanshu Maddhesiya(A12/11811293)

(3.)Deepak Mewada(A28/11807693)

(4.)Ravishekhar Singh(A21/11807671)

Submitted To – Dr.Rahul Sir

Contents –

1. Introduction to project
2. What is personal voice assistant ?
3. Start Programming
4. Python external package requirements
5. Create helpful functions
6. Create the main program
7. Program
8. Implementation

9.) Conclusion

INTRODUCTION TO PROJECT-

Who doesn't want to have the luxury to own an assistant who always listens for your call, anticipates your every need, and takes action when necessary? That luxury is now available thanks to artificial intelligence-based voice assistants.

Voice assistants come in somewhat small packages and can perform a variety of actions after hearing your command. They can turn on lights, answer questions, play music, place online orders and do all kinds of AI-based stuff.

Voice assistants are not to be confused with virtual assistants, which are people who work remotely and can, therefore, handle all kinds of tasks. Rather, voice assistants are technology based. As voice assistants become more robust, their utility in both the personal and business realms will grow as well.

What is personal voice assistant -

A voice assistant or intelligent personal assistant is a software agent that can perform tasks or services for an individual based on verbal commands i.e. by interpreting human speech and respond via synthesized voices. Users can ask their assistants’ questions, control home automation devices, and media playback via voice, and manage other basic tasks such as email, to-do lists, open or close any application etc with verbal commands.

Let me give you the example of Braina (Brain Artificial) which is an intelligent personal assistant, human language interface, automation and voice recognition software for Windows PC. Braina is a multi-functional AI software that allows you to interact with your computer using **voice commands** in most of the languages of the world.

Braina is a multi-functional AI software that allows you to interact with your computer using voice .

Start programming-

I will start by stating what I want this program to do. This program will run specific commands once a wake word is given by the users voice. This program will then execute those commands audibly back to the user.

The commands to be executed are as follows:

1.Say a random greeting back to the user, if the user used a greeting word.

2.Get the date for the user

3.Get the time for the user

4.Get information about a person to the user

(NOTE: We can use Wikipedia for this)

For example, the user may say ‘hey computer, what time is it?’ or ‘okay computer, what is today’s date ?’ and the virtual assistant will respond accordingly.

We need to install a few packages pyadio SpeechRecognition gTTSand wikipedia .

pip install pyaudio  
pip install SpeechRecognition   
pip install gTTS  
pip install Wikipedia.

Python external package requirements –

-> gTTS – Google Text To Speech, for converting the given text to speech

-> speech\_recognition – for recognizing the voice command and converting to text

-> selenium – for web based work from browser

-> wolframalpha – for calculation given by user

-> playsound – for playing the saved audio file.

-> pyaudio – for voice engine in python We will use the warnings library to ignore the warnings we get with this program. The library speech\_recognition will be used to recognize speech.

The os library will allow us to interact with the Operating System. The gtts library will help us convert text to speech. The wikipedia library will allow us to get information about a person from Wikipedia. The datetime library will allow us to get the current date and time. The calendar library will allow us to get the day of the week, and the random library will be used for randomization.

-> import os

-> from gtts import gTTS

-> import datetime

-> import warnings

-> import calendar

-> import random

-> import wikipedia

We will ignore any warning messages that may be given during the execution of this program.

# Ignore any warning messages

warnings.filterwarnings('ignore')

Create helpful function –

We will start creating helpful functions that will make the code look clean and execute certain commands.

First we need a function that can take in audio (a voice command) and recognize the speech, then return that speech as a string (text). Let’s call this function recordAudio().

**# Record audio and return it as a string**  
def recordAudio():   
**# Record the audio**  
r = sr.Recognizer()  
with sr.Microphone() as source:   
print('Say something!')  
audio = r.listen(source)  
  
**# Speech recognition using Google's Speech Recognition**data = ''  
try:  
data = r.recognize\_google(audio)  
print('You said: ' + data)  
except sr.UnknownValueError:  
print('Google Speech Recognition could not understand')  
except sr.RequestError as e:  
print('Request error from Google Speech Recognition')

return data

Perfect ! We now have a function to record the audio. Let’s create a function for the program to respond back to the user audibly and call it assistantResponse(). The function will take in a string (text) and convert it to audio. I will also have this function print the text to the screen for testing purposes.

**# Function to get the virtual assistant response**  
def assistantResponse(text):  
print(text) **# Convert the text to speech**  
myobj = gTTS(text=text, lang='en', slow=False)  
  
**# Save the converted audio to a file**   
myobj.save('assistant\_response.mp3') **# Play the converted file**  
os.system('start assistant\_response.mp3')

Next, we will create a function to take in some text and check if the wake word was given in that text. For this program I created two wake words “okay computer” and “hey computer”, similar to the Google assistant wake word “hey Google” and “okay Google”. If the wake word was detected from the text then the function will return True otherwise it will return False.

**# A function to check for wake word(s)**  
def wakeWord(text):  
WAKE\_WORDS = ['hey computer', 'okay computer']   
text = text.lower() **# Convert the text to all lower case words # Check to see if the users command/text contains a wake word**   
for phrase in WAKE\_WORDS:  
if phrase in text:  
return True **# If the wake word was not found return false**  
return False

Now, I will create a function to return today's date as a string. For example it will return “Today is Tuesday April the 7th” if today was indeed Tuesday April 7th . Let’s call this function getDate().

def getDate():  
  
now = datetime.datetime.now()  
my\_date = datetime.datetime.today()  
weekday = calendar.day\_name[my\_date.weekday()]# e.g. Monday  
monthNum = now.month  
dayNum = now.day  
month\_names = ['January', 'February', 'March', 'April', 'May',  
'June', 'July', 'August', 'September', 'October', 'November',   
'December']  
ordinalNumbers = ['1st', '2nd', '3rd', '4th', '5th', '6th',   
'7th', '8th', '9th', '10th', '11th', '12th',   
'13th', '14th', '15th', '16th', '17th',   
'18th', '19th', '20th', '21st', '22nd',   
'23rd', '24th', '25th', '26th', '27th',   
'28th', '29th', '30th', '31st']  
  
return 'Today is ' + weekday + ' ' + month\_names[monthNum - 1] + ' the ' + ordinalNumbers[dayNum - 1] + '.'

Next, we will create a function that takes in text and returns a random greeting response as text to the user, if the user said a greeting input like ‘hello’ or ‘hi’ for example the function will return some random greeting like ‘howdy’. This function will be called greeting().

**# Function to return a random greeting response**  
def greeting(text):  
**# Greeting Inputs**  
GREETING\_INPUTS = ['hi', 'hey', 'hola', 'greetings', 'wassup', 'hello'] **# Greeting Response back to the user**  
GREETING\_RESPONSES = ['howdy', 'whats good', 'hello', 'hey there'] **# If the users input is a greeting, then return random response**  
for word in text.split():  
if word.lower() in GREETING\_INPUTS:  
return random.choice(GREETING\_RESPONSES) + '.' **# If no greeting was detected then return an empty string**  
return ''

Last but not least we will create a function to get a person’s first and last name from text after detecting the key command, ‘who is’. Once we detect the word ‘who’ followed by the word ‘is’ then we will return the next two words as a single string (the next two words should be the person’s first name followed by that person’s last name).

For example if the user says ‘Who is LeBron James ?’ , then the function will take in that text and simply return ‘LeBron James’ . We will use this function later on to get a two sentence summary about the person from Wikipedia. This function will be called getPerson().

**# Function to get a person first and last name**  
def getPerson(text):  
wordList = text.split()# Split the text into a list of words for i in range(0, len(wordList)):  
if i + 3 <= len(wordList) - 1 and wordList[i].lower() == 'who' and wordList[i + 1].lower() == 'is':  
return wordList[i + 2] + ' ' + wordList[i + 3]

Create the main program-

All of the most popular virtual assistants (Google Assistant, Amazon Alexa, & Apples Siri) continuously listen to your conversation and waits to execute commands only after hearing the wake word like ‘Okay Google’, ‘Alexa’ , or ‘Hey Siri’.

This means we will need to have the program consistently listening for the wake word, so we will need a continuous loop that runs forever recording the audio.

Once the wake word is said, then we will check to see if the user said a greeting, ‘date’, ‘time’, or ‘who is’ , and have the computer respond accordingly by audio.

while True:  
**# Record the audio**  
text = recordAudio()  
response = '' **#Empty response string**  
  
**# Checking for the wake word/phrase**  
if (wakeWord(text) == True):  
**# Check for greetings by the user**  
response = response + greeting(text) **# Check to see if the user said date**  
if ('date' in text):  
get\_date = getDate()  
response = response + ' ' + get\_date **# Check to see if the user said time**  
if('time' in text):  
now = datetime.datetime.now()  
meridiem = ''  
if now.hour >= 12:  
meridiem = 'p.m' **#Post Meridiem (PM)**  
hour = now.hour - 12  
else:  
meridiem = 'a.m'**#Ante Meridiem (AM)**hour = now.hour **# Convert minute into a proper string**  
if now.minute < 10:  
minute = '0'+str(now.minute)

else:  
minute = str(now.minute) response = response + ' '+ 'It is '+ str(hour)+ ':'+minute+' '+meridiem+' .'  
  
**# Check to see if the user said 'who is'**  
if ('who is' in text):  
person = getPerson(text)  
wiki = wikipedia.summary(person, sentences=2)   
response = response + ' ' + wiki  
  
**# Assistant Audio Response**  
assistantResponse(response)

Program-

Implementation-

Conclusion-

what the future holds?

Throughout the history of computing, user interfaces have become progressively natural to use. The screen and keyboard were one step in this direction. The mouse and graphical user interface were another. Touch screens are the most recent development. The next step will most likely consist of a mix of augmented reality, gestures and voice commands. After all, it is often easier to ask a question or have a conversation than it is to type something or enter multiple details in an online form.

The more a person interacts with voice-activated devices, the more trends, and patterns the system identifies based on the information it receives. Then, this data can be utilized to determine user preferences and tastes, which is a long-term selling point for making a home smarter. Google and Amazon are looking to integrate voice-enabled artificial intelligence capable of analyzing and responding to human emotion.