Estudiante: Sara Cristina Prada Medina. **Código:** 20201186762. **Programa:** Ing. De Software.
Clase: Data Science I. **Tema:** Voice Recognition. **Trabajo:** voiceAssistant APP.

Voice Recognition – voiceAssistant APP

Descripción de la aplicación:

Se realizó un simple y limitado asistente de voz que puede ser ejecutado y permanecer corriendo hasta ser llamado mediante un comando clave ("sol"), el cual activará explícitamente al asistente. Este preguntará al usuario que tarea desea ejecutar y, de acuerdo a la respuesta del usuario, el asistente realizará la acción indicada. El programa solo podrá realizar las siguientes tareas:

- Crear un documento txt.
- Leer un documento txt.
- Abrir el navegador.

Pre-requisitos:

Se deben instalar tres dependencias para poder hacer funcionar el programa. Estos son:

- Speechrecognition: Para hacer posible el reconocimiento de voz.
- Pyaudio: Para hacer uso del micrófono.
- Pyttsx3: Para hacer posible la transformación de texto a voz.

Requisitos:

Se deben importar las siguientes dependencias:

- Speechrecognition.
- Pyttsx3.
- Webbrowser: Para poder gestionar el navegador.

Funciones:

Para que el programa trabaje correctamente, se ha creado las siguientes funciones:

- ***Hablar (speak):***

  Esta función es para indicarle al asistente qué debe decirle al usuario de forma verbal.

- ***Calibrar (calibrar):***

  El objetivo de esta función es evaluar el ruido ambiental en donde se encuentra el usuario y ajustar la sensibilidad del reconocedor de voz de acuerdo a lo detectado.

- ***Audio a Texto (audioToText):***

  Esta función permite pasar el contenido dictado por el usuario a texto. Según lo que diga el usuario, el asistente podrá – dentro de sus limitaciones – realizar las tareas indicadas o mencionar su incapacidad para realizar dichas tareas.

- ***Crear archivo (createFile):***

  Con esta función, el asistente le será posible crear un archivo de texto teniendo en cuenta los comentarios del usuario respecto al nombre y contenido que deberá tener dicho archivo.

  De no entender un nombre para el documento, el asistente le indicará al usuario de este hecho y le pedirá que lo repita una vez se haya cargado el programa nuevamente.

  Por otro lado, si el asistente no entiende cual es el contenido dictado por el usuario que deberá tener el archivo nuevo, le informará que simplemente se ha creado un documento txt vacío.

- ***Leer archivo (readFile):***

  Esta función permitirá al asistente leer cualquier archivo de texto que el usuario le indique. Si no encuentra ningún archivo con ese nombre en la carpeta donde el asistente está ubicado, le informará al usuario que tal documento no existe.

- ***Abrir navegador (openBrowser):***

  Con esta función, le será posible al asistente abrir el navegador predeterminado del usuario y redirección la página al buscador Google cuando así lo indique el usuario.

- ***Usar micrófono (useMic):***

  El objetivo de esta función es activar nuevamente el reconocimiento de voz para poder ejecutar las tareas de crear archivo y leer archivo de forma adecuada.

  Aquí se llama también a la función de calibrar con el objetivo de realizar los cambios pertinentes a la sensibilidad del reconocedor frente a las posibles variaciones en el ruido ambiental que puedan existir entre el momento de llamar al asistente y el momento de pedir crear o leer un documento.

Estudiante: Sara Cristina Prada Medina. **Código:** 20201186762. **Programa:** Ing. De Software.
Clase: Data Science I. **Tema:** Voice Recognition. **Trabajo:** voiceAssistant APP.

Detalles menores:

- Se ha programado la aplicación de forma que pueda informarle al usuario de cuando una tarea terminó y cuando la aplicación está lista para reconocer el audio del usuario nuevamente.

- Se ha dejado líneas para imprimir en la terminal las palabras detectadas por la aplicación tras transformar lo dicho por el usuario en texto con el fin de utilizarlo como feedback visual para dicho usuario.

Desempeño:

El programa funciona adecuadamente, aunque hay ocasiones en las que no detecta toda la frase o detecta el contenido incorrecto. Sin embargo, lo anterior puede deberse a que el micrófono integrado del portátil de la presente autora del trabajo posee un bajo volumen. Este micrófono ya está configurado para presentar el máximo volumen, pero, a pesar de ello, el problema de bajo sonido captado del micrófono sigue presentándose y no ha habido forma de arreglar tal situación hasta el momento.

También es importante mencionar que el programa puede llegar a tener latencia en cuanto al procesamiento del audio y la respuesta esperada. No es claro si esto se debe a las bajas especificaciones del portátil de la autora o a que siempre se realiza un proceso de calibración de sensibilidad del reconocedor de voz cada vez que se le pide al usuario algún dato, el subsecuente proceso de transformar el audio captado a texto por el programa luego de dicha calibración también puede influir de alguna forma.

A pesar de lo anterior, este simple asistente de voz se ejecuta correctamente y puede ser utilizado si así alguien lo desea y ve útil o práctico las tareas que este es capaz de realizar, dado que el usuario no tiene que hacer nada, solo activar el asistente y hacer que este se encargue de ello.

Prueba de la aplicación:

https://youtu.be/QV4cSeQ6-xk

Estudiante: Sara Cristina Prada Medina. **Código:** 20201186762. **Programa:** Ing. De Software.
Clase: Data Science I. **Tema:** Voice Recognition. **Trabajo:** voiceAssistant APP.

Código de la aplicación:

```
# ----------------- REQUISITOS --------------------------

## DEPENDENCIAS NECESARIAS

## pip install speechrecognition

## pip install pyaudio

## pip install pyttsx3

import speech_recognition as sr

import pyttsx3 as tts

import webbrowser

# ------------------ FUNCIONES ------------------------

# rate: velocidad al hablar

def speak(text):

    engine.setProperty('rate', 140)

    engine.say(text)

    engine.runAndWait()

# Calibrar detalles

def calibrar():

    r.adjust_for_ambient_noise(source, duration=0.8) # Calibra el umbral de energía (sensibilidad del reconocedor) según el ruido ambiental detectado.

    r.pause_threshold = 0.8 # Mínima cantidad de silencio que registrará como final de la frase.
```

```
# Pasar el audio en texto
def audioToText(sound):
    texto = r.recognize_google(sound, language='es-ES')
    texto = texto.lower()
    return texto


# crear archivo de texto
def createFile():
    fileName = useMic('Nombre el documento: ')
    if fileName[0] == 0:
        with open(f"{fileName[1]}.txt", "w") as f:
            fileContent = useMic('Díctame el contenido del documento: ')
            if fileContent[0] == 0:
                f.write(f"{fileContent[1]}")
                speak('Archivo de texto creado.')
            else:
                speak(f'{fileContent[1]}. Se ha creado un archivo de texto sin contenido.')
    else:
        speak(f'{fileContent[1]}. Inténtalo nuevamente.')


# leer contenido de un documento
def readFile():
    fileName = useMic('Nombre del documento a leer: ')
    if fileName[0] == 0:
```

```
        try:
            textInFile = open(f'{fileName[1]}.txt', "r")
            lines = textInFile.readlines()
            for line in lines:
                speak(line)
        except:
            speak(f'No exite un documento txt con el nombre {fileName[1]}')
    else:
        speak(f'{fileName[1]}. Inténtalo nuevamente.')


# abrir navegador
def openBrowser():
    url='https://www.google.co'
    webbrowser.open(url)
    speak('Se ha abierto el navegador y se ha redireccionado a la página de Google.')


# validar audio para las funciones anteriores
def useMic(line):
    with sr.Microphone() as source:
        calibrar()
        speak(f'{line}')
        audio = r.listen(source)
        try:
            text = audioToText(audio)
            print("Dijiste: {}".format(text))
```

```
            flag = 0
        except:
            text = "Lo siento, no pude entenderte."
            flag = 1
        return flag, text


# ---------- INICIAR PROGRAMA ----------


# Configurar voz
engine = tts.init()
voices = engine.getProperty('voices') # Información sobre la voz de un sintetizador de voz
engine.setProperty('voice', voices[2].id) # Cambiar voz
r = sr.Recognizer() # Iniciar reconocimiento


intro = 0 # para confirmar al usuario el inicio del programa


with sr.Microphone(device_index=1) as source:
    while True:
        calibrar()
        if intro == 0:
            speak('Programa cargado')
            intro = 1
        audio = r.listen(source)
        try:
            text = audioToText(audio)
```

```
        print("Dijiste: {}".format(text))
        if "sol" in text:
            calibrar()
            speak('Hola, ¿cómo puedo ayudarte?')
            audio = r.listen(source)
            try:
                text = audioToText(audio)
                print("Dijiste: {}".format(text))
                if text == "crear texto":
                    createFile()
                elif text == "leer texto":
                    readFile()
                elif text == "abrir navegador":
                    openBrowser()
                elif text == "detente":
                    speak('Adios...')
                    break
                else:
                    speak('Lo siento, no puedo ayudarte con eso.')
            except:
                speak('Lo siento, no puedo entenderte')
            intro = 0
            speak('Espere a que el programa se cargue nuevamente.')
    except:
        continue
```