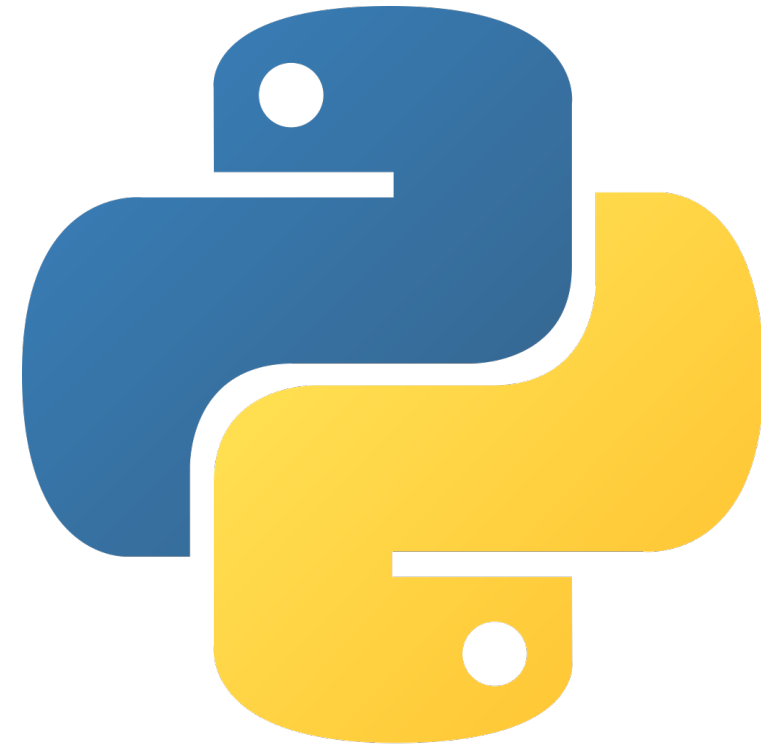


RECONOCIMIENTO DE VOZ CON PYTHON

Realizado por: Moisés Noguera Carrillo



Índice

Instalación

Paquete SpeechRecognition

Paquete PyAudio

Funcionamiento

Demo

Instalación

Para poder hacer uso de cualquiera de los paquetes de reconocimiento de voz que ofrece Python, en primer lugar es necesario instalar Python (versión 3.3+) y el gestor de paquetes de Python llamado Pip:

- Instalar Python: En macOS y Linux viene instalado por defecto. En Windows, acceder a la página web oficial de Python y descargar la última versión disponible.
- Instalar Pip:
 - Descargar el script `get-pip.py`
 - En un terminal ejecutar el comando: `python get-pip.py` (macOS y Linux) o `py get-pip.py` (Windows).

Instalación

Para instalar el paquete de reconocimiento de voz basta con ejecutar la orden:

- `pip install SpeechRecognition`

El paquete SpeechRecognition

Python ofrece una gran variedad de paquetes de reconocimiento de voz, sin embargo, SpeechRecognition destaca por su facilidad de uso.

Para reconocimiento de voz es necesario recibir una entrada de audio y SpeechRecognition permite captar las entradas de audio de forma inmediata y fácil sin necesidad de construir scripts para acceder a los micrófonos.

El paquete proporciona una serie de APIs para realizar el reconocimiento de voz, aunque para este ejemplo se va a acceder a la Google Web Speech API.



```
import speech_recognition as sr

# Crear una instancia de la clase Recognizer.
r = sr.Recognizer()

# Instancia de la clase AudioFile
harvard = sr.AudioFile("harvard.wav")
with harvard as source:
    # Capta los datos del archivo de audio en una variable de tipo AudioData
    audio = r.record(source)

# Mostrar por pantalla el audio transcrito
print(r.recognize_google(audio))
```

El paquete SpeechRecognition: Clase Recognizer

Recognizer es la clase principal de SpeechRecognition mediante la cual se realiza el reconocimiento de voz. Una instancia de esta clase dispone de siete métodos diferentes para reconocer voz desde una fuente de audio usando distintas APIs. Como se ha comentado anteriormente, en el ejemplo actual se va a usar la Google Web Speech API.

El argumento que recibe *recognize_google()* debe ser de tipo *AudioData*, que se obtiene a través de un archivo de audio o mediante el audio captado por un micrófono.

El paquete PyAudio

Para poder reconocer audio captado por un micrófono hay que instalar el paquete PyAudio:

En Debian Linux: `sudo apt-get install python-pyaudio python3-pyaudio`

En macOS: `brew install portaudio` y `pip install pyaudio`

En Windows: `pip install pyaudio`



```
import speech_recognition as sr


# Crear una instancia de la clase Recognizer y Microphone.
r = sr.Recognizer()
mic = sr.Microphone()

with mic as source:
    audio = r.listen(source)

print(r.recognize_google(audio))
```

El paquete PyAudio: Clase Microphone

Una vez instalado PyAudio, se puede hacer uso de la clase *Microphone* de *SpeechRecognition* para captar audio directamente desde el micrófono. Para ello, la clase proporciona un método llamado *listen()* que recibe al igual que *record()* en el ejemplo anterior un argumento de tipo *AudioFile*. Como salida proporciona una instancia de la clase *AudioData*.



FUNCIONAMIENTO: FUNCIÓN RECORD_AUDIO

```
# Function to record audio from microphone
def record_audio(ask = False):
    # Use the microphone as source of audio
    with sr.Microphone() as source:
        if ask:
            jarvis_speak(ask)
        audio = r.listen(source, phrase_time_limit=4)
        voice_data = ''
        try:
            voice_data = r.recognize_google(audio, language='es-ES')
        except sr.UnknownValueError:
            jarvis_speak('Lo siento, no te he entendido...')
        except sr.RequestError:
            jarvis_speak('Lo siento, servicio no disponible por el momento')
    return voice_data
```



FUNCIONAMIENTO: FUNCIÓN JARVIS_SPEAK



```
# Function to make Jarvis speak
def jarvis_speak(audio_string):
    tts = gTTS(text=audio_string, lang='es')
    r = random.randint(1, 1000000)
    audio_file = 'audio-' + str(r) + '.mp3'
    tts.save(audio_file)
    playsound.playsound(audio_file)
    print(audio_string)
    os.remove(audio_file)
```

FUNCIONAMIENTO: FUNCIÓN RESPOND

```
# Voice commands
def respond(voice_data):
    if 'dime tu nombre' in voice_data:
        jarvis_speak('Mi nombre es Yarvis y soy tu asistente virtual')

    if 'qué hora es' in voice_data:
        now = datetime.datetime.now()
        hour = '{:02d}'.format(now.hour)
        minute = '{:02d}'.format(now.minute)
        jarvis_speak('Son las ' + hour + ':' + minute)

    if 'fecha' in voice_data:
        now = datetime.datetime.now()
        month_day = '{:02d}'.format(now.day)
        month_name = now.strftime("%B")
        month_name = month_spanish(month_name)
        year = '{:02d}'.format(now.year)
        week_day = now.strftime("%A")
        week_day = week_day_spanish(week_day)
        jarvis_speak('Es ' + week_day + ', ' + month_day + ' de ' + month_name + ' de ' + year)

    if 'buscar' in voice_data:
        search = record_audio('¿Qué quieres que busque?')
        url = 'https://google.com/search?q=' + search
        webbrowser.get().open(url)
        jarvis_speak('Esto es lo que he encontrado sobre ' + search)

    if 'localización' in voice_data:
        location = record_audio('¿Qué localización deseas buscar?')
        url = 'https://google.es/maps/place/' + location + '/&'
        webbrowser.get().open(url)
        jarvis_speak('Aquí está la localización de ' + location)

    if 'hasta luego' in voice_data:
        jarvis_speak('¡Adiós!')
        exit()
```



DEMO: JARVIS, UN ASISTENTE DE
VOZ SENCILLO.