Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Manual de instalación para el prototipo de asistente virtual para el laboratorio de procesamiento de audio con herramientas de código abierto.

IS - 2024

Ezequiel Esquivel Núñez - B82796

Guía paso a paso para la instalación y uso del proyecto

Profesor: Marvin Coto Jiménez

Índice

1. Introducción

En este documento se puede encontrar una descripción paso a paso de como hacer funcionar el asistente virtual en cualquier computadora con sistema operativo Linux, la razón de esto es para poder utilizar herramientas de código abierto, algunas de ellas que podemos encontrar a lo largo del proyecto son Whisper, CoquiTTS, openAI y Speechbrain, todas estas son base fundamental para el debido funcionamiento del proyecto.

En esta guía se encontrará con **3 secciones** distintas. La **sección 2** es una guía sobre como instalar todas las librerías necesarias para que el proyecto funcione, se debe de instalar todo lo que viene en esta sección.

En la **sección 3** se encuentra una explicación sobre como funciona el proyecto y qué hacen los archivos fundamentales para el funcionamiento del mismo.

Finalmente en la **sección 4** podrá encontrar el link de descarga del proyecto, así como una guía paso a paso sobre como ejecutarlo. Se recomienda también entender el código de cada archivo para tener un mejor manejo del mismo, así como tener *Visual Studio* instalado.

A continuación se le presenta la posibilidad de seguir el manual de manera guiada mediante un video, para acceder al video haga click aquí.

2. Instalación de librerías y programas necesarios para el proyecto

Es obligatorio tener python3 en la versión 3.9.5 o más para el funcionamiento del proyecto, así como Numpy en la versión 1.22. Para revisar estas versiones puede hacer lo siguiente:

- python3 –version
- `python3 -c "import numpy as np; print(np.__version__)"`

  En caso de que no sean las indicadas o una versión más reciente, se le insta a actualizar las versiones.

2.1. Instalación de Whisper

Whisper es el programa encargado de transcribir la voz a texto, este recibe los audios generados por *grabar.sh* y los transcribe a texto dentro del programa *identifier.py*.

2.1.1. ¿Como instalarlo?

Ingrese lo siguiente en la terminal y presione enter:

```
pip install whisper
```

2.2. Instalación de CoquiTTS y mpg321

Estas dos librerías trabajan en conjunto para transcribir de texto a voz, son las encargadas de comunicar mediante voz a la persona de lo que está pasando, así como de darle voz a las respuestas generadas por chatgpt, esta librería es utilizada dentro de *identifier.py* y *main.py*.

2.2.1. ¿Cómo instalarlos?

Primeramente ejecute la instalación de coquiTTS:

```
pip install tts
```

Seguidamente para instalar mpg321:

```
pip install mpg321
```

2.3. Instalación de speechbrain

Speechbrain es la herramiente encargada del reconocimiento de hablante, se utiliza dentro de *identifier.py*.

2.3.1. ¿Como instalarlo?

Ejecute:

```
pip install speechbrain
```

2.4. Instalación de OpenAI

Este es el motor de búsqueda, es el encargado de recibir el texto transcrito de la voz como entrada al motor de búsqueda, procesarlo y dar una respuesta la cual es transcrita a voz mediante *CoquiTTS*, esta librería es utilizada en *identifier.py* y *main.py*.

2.4.1. ¿Cómo instalarlo?

Ejecute:

```
pip install openai
```

2.5. ¿Qué hacer si en el proceso de instalación hay errores?

Los errores van a depender de muchos aspectos intrínsecos del computador, así como de la versión instalada de Linux, estos no suelen aparecer, en realidad son bastante raros, pero en caso de aparecer y tener un efecto directo sobre el funcionamiento del proyecto, se insta a la búsqueda de una solución en internet.

2.6. Otros

A continuación se mostrarán algunas otras instalaciones que ya están por defecto en Linux, pero que nunca está de más ejecutar en caso de que no lo estuvieran. Si ya están instaladas, es probable que el comando de error, pero esto no es problema. Se recomienda encarecidamente ejecutar cada uno de los siguintes comandos:

```
pip install config
```

```
pip install typer
```

```
pip install requests
```

```
pip install glob
```

```
pip install os
```

```
pip install time
```

```
pip install subprocess
```

```
pip install torchaudio
```

3. ¿Como funciona el proyecto?

Aunque el proyecto que se descarga en la sección 4 contengan varios archivos, solo es necesario ejecutar 2 de manera simultánea para que todo el proyecto funcione.

3.1. Permisos de ejecución

Para poder ejecutar los archivos es necesario que mediante una terminal acceda a la carpeta llamada *Main* y ejecute el siguiente comando:

```
chmod +x <nombre_del_archivo>
```

Este comando anterior otorga permisos de ejecución a todo, debe de otorgarle estos permisos a los archivos llamados: *grabar.sh*, *identifier.py*, *main.py*, *config.py*, *temp_audio.wav* *iniciar_grabar.sh* y *iniciar_identifier.sh*

3.2. grabar.sh

Este archivo no requiere de ninguna librería para funcionar, solo necesita de una entrada de audio para poder grabar. Este código genera archivos con extensión .wav, la razón de esto es que la librería speechbrain funciona mejor con archivos que guarden mayor cantidad de información.

3.2.1. Ejemplo de ejecución de grabar.sh

Para ello acceda mediante la terminal a la ruta donde está almacenado el archivo, para ello puede utilizar el comando *cd* e ir desplazandose entre los folders hasta llegar a la carpeta que lo contiene. Para saber si está en la carpeta correcta puede utilizar el comando *ls* y verificar si **grabar.sh** se encuentra allí. Una vez aquí se debe de ingresar en la terminal lo siguiente: **./grabar.sh**. Una vez hecho esto debería de desplegarse texto que indica que se está grabando en tiempo real audio, para parar la grabación se debe de presionar **ctrl + z**.
Grabar.sh funciona de manera que cuando haya silencio corta la grabación que estaba en proceso, guarda la información en el archivo.wav y luego empieza a grabar uno nuevo hasta tener otro silencio. Este archivo es la base del funcionamiento del proyecto.

3.2.2. Resumen sobre grabar.sh

- Se encarga de grabar audio en formato .wav
- Para ejecutarlo acceda mediante la terminal a la ruta donde se encuentra el archivo.
- Ingrese en la terminal dentro de la ruta lo siguiente: **./grabar.sh**

3.3. identifier.py

Este programa se encarga de recibir los audios creados por **grabar.sh** de la sección anterior y transcribirlos a texto mediante el uso de la librería *Whisper*, cuando la traducción sea la de *abrir chat* se activa el reconocimiento de hablante por parte de la librería *speechbrain*, es decir, compara la entrada de audio que decía la palabra clave con unos archivos de referencia que se encuentran dentro de la carpeta llamada **Embeddings**, es en esta carpeta donde se deben de guardan los audios de referencia, los cuales deben decir *abrir chat*. es ideal que el nombre de este archivo lleve consigo el nombre de la persona. Si la librería speechbrain detecta que el archivo de audio generado por *grabar.sh* coincide con la referencia, se da paso a la ejecución del archivo llamado *main.py*.

3.3.1. ¿Cómo ejecutar identifier.py?

Abra una terminal y acceda a la ruta donde se encuentra el archivo y ejecute lo siguiente: **python3 identifier.py**

3.3.2. Resumen sobre identifier.py

- Se encarga de detectar la palabra clave, identificar al hablante y ejecutar a chatgpt en caso de coincidir.
- Para ejecutarlo acceda mediante la terminal a la ruta y ejecute: **python3 identifier.py**

4. Guía paso a paso para hacer funcionar el proyecto

4.1. Descargar el proyecto

Descargue el proyecto haciendo click aquí, en el recuadro verde que dice *code* puede clonar el repositorio, o bien, descargarlo en forma comprimida:

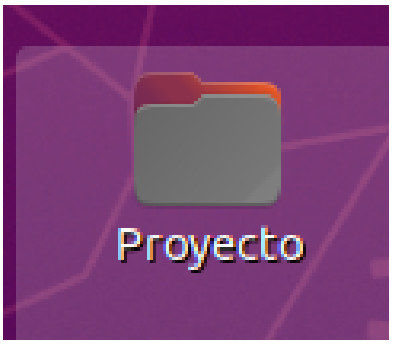

Figura 1: Resultado obtenido al descomprimir el proyecto

Dentro de esta carpeta se encuentran varias carpetas más, la realmente importante es la que se llama *Main.py*.

4.2. Instalación de librerías

Antes de poder proseguir con el proyecto es necesario que haya instalado todas las librerías, el procedimiento fue explicado anteriormente en este documento.

4.3. Otorgar permisos de ejecución

Debe seguir el procedimiento explicado en la sección 3.1 del documento, a continuación se presenta un ejemplo:

Para acceder a la ruta:

```
cd Desktop/Proyecto_AI/Proyecto/Main
```

Para dar permisos de ejecución:

```
chmod +x grabar.sh
```

si precisa conocer la ruta exacta puede utilizar el comando *pwd*, el cual le brinda en pantalla la ruta de donde se encuentra ubicado actualmente.

4.4. Ejecución simultánea de *identifier.py* y *grabar.sh*

Una vez haya otorgado permisos de ejecución a todos los programas necesarios y haya instalado todas las librerías es posible ejecutar el proyecto, para ello sigue los siguientes pasos:

1. Abra dos terminales.
2. En ambas terminales acceda a la ruta de la carpeta del proyecto llamada *Main*.
3. En una de las terminales escriba *./grabar.sh* pero aún no lo ejecute.
4. En la otra terminal escriba *python3 identifier.py*.
5. Una vez tiene ambos comandos listos, ejecute primero *./grabar.sh* y luego *python3 identifier.py*
6. Diga la frase **abrir chat**, la traducción a texto de lo que se acaba de decir debe de ser visible en la terminal que está ejecutando al programa *identifier.py*, si accede al código puede visualizar que se aceptan varias traducciones como válidas para cuando se detecta la palabra clave.
7. Si la traducción es correcta el programa empezará a reconocer al hablante, en caso de que coincidan, se abrirá chatgpt.
8. Más adelante discutiremos como agregar más referencias de hablantes al código, pues actualmente el proyecto solo acepta como válidas las voces del profesor Marvin Coto y del estudiante Ezequiel Esquivel.

4.5. Programar inicio automático del proyecto al iniciar la computadora

Es posible automatizar el inicio del proyecto al iniciar la computadora, lo cual convierte de cierta manera el dispositivo en un módulo de asistente virtual, para ello se utilizarán 2 códigos más, los cuales son *iniciar_grabar.sh* y *iniciar_identifier.sh*. Estos dos documentos lo que hacen es abrir dos terminales, acceder en ellas a la carpeta *Main* y ejecutan los programas antes vistos. Para hacer esto siga los siguientes pasos:

1. Dentro de las aplicaciones de Linux debe de encontrar una llamada *Aplicaciones de inicio* o bien, *Startup application*, acceda a ella.

2. Se le debería desplegar una ventana parecida a la siguiente, presione click sobre *añadir* o *add*:

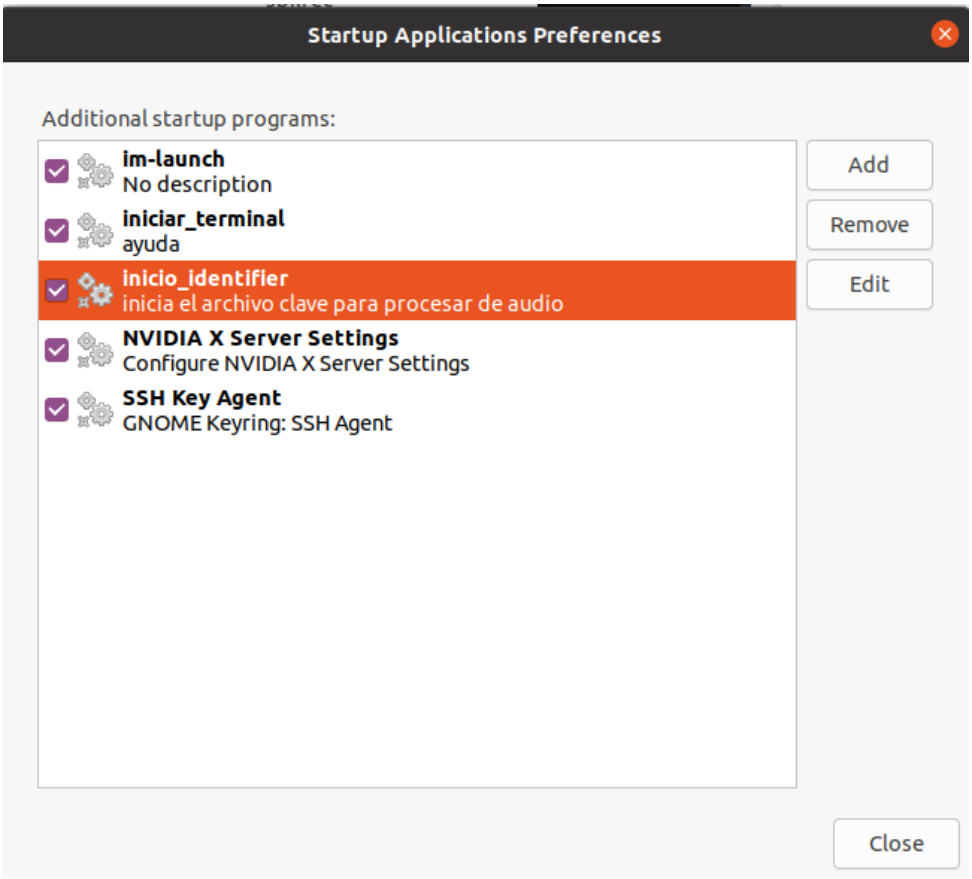

Figura 2: APlicaciones de inicio

3. Una vez hecho esto debería de desplegarse una ventana parecida a la anterior:

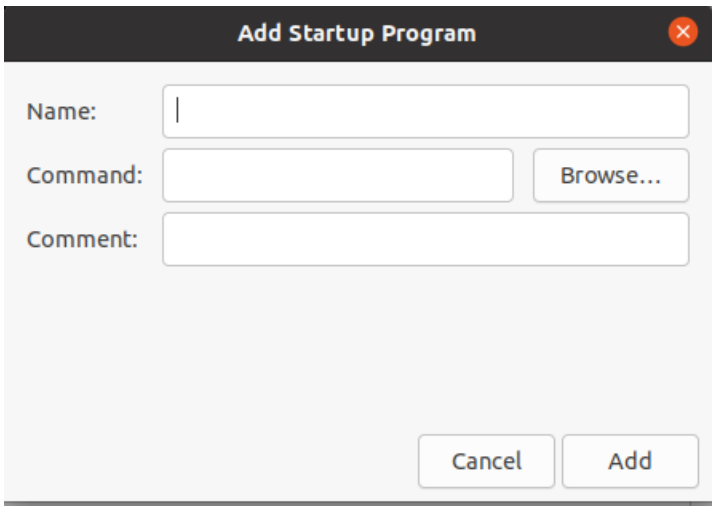

Figura 3: Ventana para la automatización del proyecto

Asígnele el nombre que guste, sin embargo, se recomienda asignarle el mismo nombre del archivo que ejecute, en la opción de *comando* presione *Browse* o *Buscar* y acceda a la carpeta *Main.py* donde se encuentra el archivo *iniciar_grabar.sh* y *iniciar_identifier.sh*, de doble click sobre uno de ellos para añadirlo a la automatización. En la sección de

comentario añada lo que guste. Finalmente de click en añadir. Repita este procedimiento para el archivo que no seleccionó, es decir, *iniciar_identifier.py* o *iniciar_grabar.sh*.

Al hacer esto, cuando inicie la computadora debería de ejecutarse el proyecto sin ningún problema.

4. Como paso adicional puede deshabilitar la contraseña de inicio de sesión, para ello siga el siguiente tutorial haciendo click aquí.

5. Análisis sobre el reconocimiento de hablante

5.0.1. instalación de audacity para crear archivos de referencia

Se recomienda utilizar *Audacity* para grabar las referencias, puede buscar un tutorial de como utilizar el programa, los archivos de audio generados deben de decir *abrir chat* y deben ser guardados en la carpeta llamada *Embeddings*. Asegúrese de que se escucha claramente el audio generado por Audacity, y que no esté cortado.

5.1. ¿Cómo agregar más personas a la lista?

Dentro del código del archivo *identifier.py* se encuentra un bucle while, cuando se detecta la palabra *abrir chat* es que se empieza con el procedimiento para reconocer al hablante, el código es el siguiente:

```
if transcript.strip() == "abrir␣chat" or transcript.strip() == "abrir␣
   ↪ chat." or transcript.strip() == "abre␣el␣chat.":

sample1 = "/home/ezequiel/Desktop/Proyecto/Proyecto/Grabaciones/
   ↪ Embeddings/marvin_sample.wav"
sample2 = audio_filename
score, prediction = verify_speakers(sample1, sample2)
print(f"Prediction:␣{prediction}")



if prediction:

    response = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input="Los␣hablantes␣coinciden.␣")
    response.stream_to_file("output.wav")
    os.system("mpg321␣output.wav")
    os.remove("output.wav")
    main_process = subprocess.Popen(["/usr/bin/python3", "main.py"])

    # Esperar a que main.py termine
    main_process.wait()

else:

    sample3 = "/home/ezequiel/Desktop/Proyecto/Proyecto/Grabaciones/
       ↪ Embeddings/ezequiel_sample.wav"
    score, prediction = verify_speakers(sample3, sample2)
```

```
    print(f"Prediction: {prediction}")

if prediction:


    response = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input="Los hablantes coinciden. ")
    response.stream_to_file("output.wav")
    os.system("mpg321 output.wav")
    os.remove("output.wav")
    main_process = subprocess.Popen(["/usr/bin/python3", "main.py"])

    # Esperar a que main.py termine
    main_process.wait()



else:
    response = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input="Los hablantes no coinciden. Prosigo a seguir buscando la
      ↪ palabra clave. ")
    response.stream_to_file("output.wav")
    os.system("mpg321 output.wav")
    os.remove("output.wav")
```

Se recomienda entender el código anterior, así como todo el código del proyecto, sin embargo, podemos notar que en el bloque recientemente mostrado una vez se detecta la palabra clave, se compara la entrada de audio con el archivo de referencia del profesor Marvin que se encuentra en la carpeta *Embeddings*, si no coincide, es decir, el tensor de predicción es falso, se pasa a comparar con el estudiante Ezequiel, si es falso seguirá esperando por un nuevo archivo de audio que diga la palabra clave para comparar. Para implementar un nuevo referente simplemente se debe agregar un else y un if antes del else que indica que no coinciden los hablantes, en el else se debe de implementar lo siguiente:

```
sampleX = "/home/ezequiel/Desktop/Proyecto/Proyecto/Grabaciones/
  ↪ Embeddings/audio_de_referencia.wav"
score, prediction = verify_speakers(sampleX, sample2)
print(f"Prediction: {prediction}")
```

El nombre de la variable *sample* debe ser distinto a los anteriores, por lo que la letra **X** puede ser cualquier otra cosa. La ruta probablemente sea distinta dependiendo de la computadora en la que se ejecute.

Seguidamente el if debe llevar lo siguiente:

```
if prediction:


response = client.audio.speech.create(
model="tts-1",
voice="alloy",
input="Los hablantes coinciden. ")
```

```
response.stream_to_file("output.wav")
os.system("mpg321 output.wav")
os.remove("output.wav")
main_process = subprocess.Popen(["/usr/bin/python3", "main.py"])

# Esperar a que main.py termine
main_process.wait()
```

De esta manera es como se implementan más referencias para activar a chatgpt.

6. Futuro del proyecto

A lo largo de este trabajo se implementa de manera exitosa un asistente virtual para el laboratorio 402, sin embargo, sigue habiendo mucho espacio para mejoras del mismo, así como modificar funciones o agregar nuevas, a continuación se presenta una lista de ellas:

- Actualmente el proyecto funciona mediante la lectura de archivos de audio creados, una mejora importante por realizar es la de hacer que sea *en vivo* la transcripción de voz a texto, esto mejoraría considerablemente problemas debido al almacenamiento de archivos.
- Actualmente la función de reconocimiento de hablante es sencilla de implementar pues la cantidad de personas que deben de ser reconocidas por *speechbrain* para ejecutar el programa es baja, sin embargo, si se piensa aumentar la capacidad del proyecto resultaría necesario implementar una base de datos con SQL o SQLite donde al detectar la palabra clave se acceda a los archivos de referencia y se compare con cada uno de ellos buscando un *tensor de predicción verdadero*, y en caso de que no haya, seguir esperando la palabra clave dicha por una persona de la base de datos.
- Es importante mejorar las entradas de audio, actualmente la traducción de voz a texto se puede ver afectada debido a la calidad de los micrófonos. Si se quiere mejorar la eficiencia con la que se traduce es importante buscar una solución sobre esto.
- Es posible implementar el reconocimiento de hablante a la hora de dialogar o hacer preguntas con ChatGPT, es decir, una vez que se detecta la palabra clave y se accede al motor de búsqueda, que el programa solamente acepte preguntas de aquellas personas que se encuentren dentro de la base de datos de SQL, o bien, de aquellas personas deseadas. Actualmente el reconocimiento de hablante funciona únicamente para la activación del proyecto, no para la conversación continua. Esta implementación probablemente se debe de hacer en el archivo *main.py*
- Actualmente cuando se le pregunta algo a chatgpt mediante voz y este responde, la pregunta se reproduce hasta el final, no hay manera de interrumpirlo, si se tuviera un feedback *en vivo* de lo que se está diciendo, y el programa estuviera escuchando al mismo tiempo que da la respuesta, es posible implementarle comandos que hagan que se detenga, o bien, conteste otra pregunta, esta mejora va de la mano con el primer punto discutido en esta sección.
- Al hacer pruebas en el laboratorio se puede observar que si los parlantes y el micrófono están muy cerca se crea una realimentación donde chatgpt se habla y pregunta a él mismo, esto es un problema que se debe de solucionar, pues este bucle infinito no pareciera ser útil.