Universidad de Costa Rica
Facultad de Ingeniería

Escuela de Ingeniería Eléctrica
IE0499 Proyecto Eléctrico
II Ciclo 2024

Reconstrucción de señales de voz utilizando algoritmos de Deep Learning

Guía paso a paso para la instalación y uso del proyecto

Profesor:
PhD. Marvin Coto

Estudiante:
Isabel Sabater Aguilar - B97037

Índice

1. Introducción

Este proyecto emplea algoritmos de Deep Learning para reconstruir señales de audio y así mejorar grabaciones afectadas por ruido. Se diseñó un experimento en el Laboratorio 402 IE, donde dos micrófonos, uno con patrón cardioide y otro con patrón omnidireccional, captaron 14 grabaciones cada uno, a diferentes distancias. Estas grabaciones fueron realizadas por dos hablantes (hombre y mujer) siguiendo un guión de 340 palabras.

El denoiser aplicado es de Resemble AI [1]. Seguidamente, para realizar las realizar las transcripciones se utiliza Deepgram [3]. Además, para el cálculo del WER se utiliza el repositorio [2]. El sistema de denoiser que se implementa para entrenamiento es de Facebook [4].

Cabe mencionar que se realiza además una identificación de hablantes basada en embeddings, un calculo de frecuencia fundamental, su varianza y la velocidad del habla promedio y su varianza.

2. Consideraciones previas a ejecutar el proyecto

Este proyecto se ejecuta desde Google Colab, preferiblemente usando una TPU o GPU para tener mayores recursos disponibles. El código de este proyecto lo puede encontrar en el repositorio de GitHub:

Reconstruccion_de_senales_de_audio

Para funcionar, inicialmente, en el entorno de Google Colab en el directorio /content/ es necesario tener las carpetas Grabaciones_de_lejos y Grabaciones_de_cerca. En estas carpetas se encuentran todas las grabaciones, solo las grabaciones captadas con el micrófono omnidireccional y solo las grabaciones captadas con el micrófono cardiode respectivamente. Los archivos .mp3 en estas carpetas tienen su formato basado en la configuración del espacio de la Figura 1.

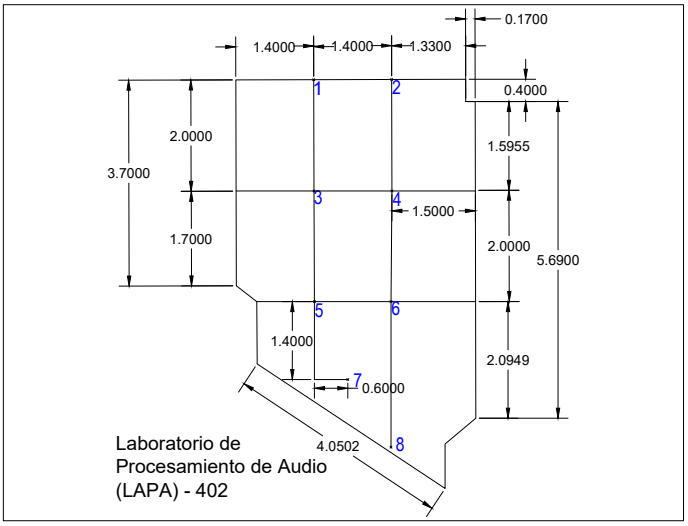


Figura 1: Configuración del espacio

Grabacion1_PuntoX_TomadaEnPuntoY_Hablante.mp3

Donde:

- X es el punto donde estaba la persona al realizarse la grabación
- Y es el punto donde estaba el micrófono que realizaba la grabación.
- Hablante es el nombre que leyó el texto durante la grabación.

Por ejemplo:

- Grabacion1_Punto4_TomadaEnPunto4_Isabel.mp3

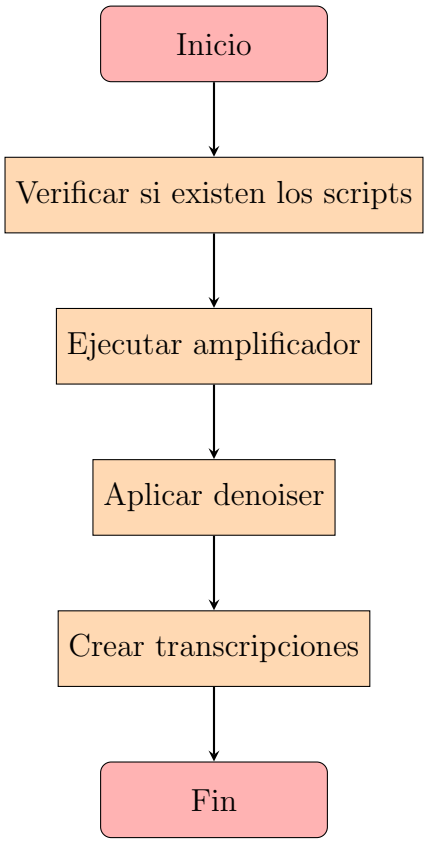
  Esta grabación la realizó Isabel, en el Punto 4 de la configuración de posiciones de la Figura 1 y fue TomadaEnPunto 4 quiere decir que se grabó con un micrófono cardiode ya que el micrófono se encontraba en el mismo punto que la persona al momento de realizar la grabación.

- Grabacion1_Punto4_TomadaEnPunto7_Isabel.mp3

  En este otro caso, si bien fue grabada por Isabel y se leyó desde el Punto 4, el micrófono estaba en el punto 7 por lo que se grabó con el micrófono omnidireccional, de lejos.

3. ¿Cómo funciona el proyecto?

3.1. main.py



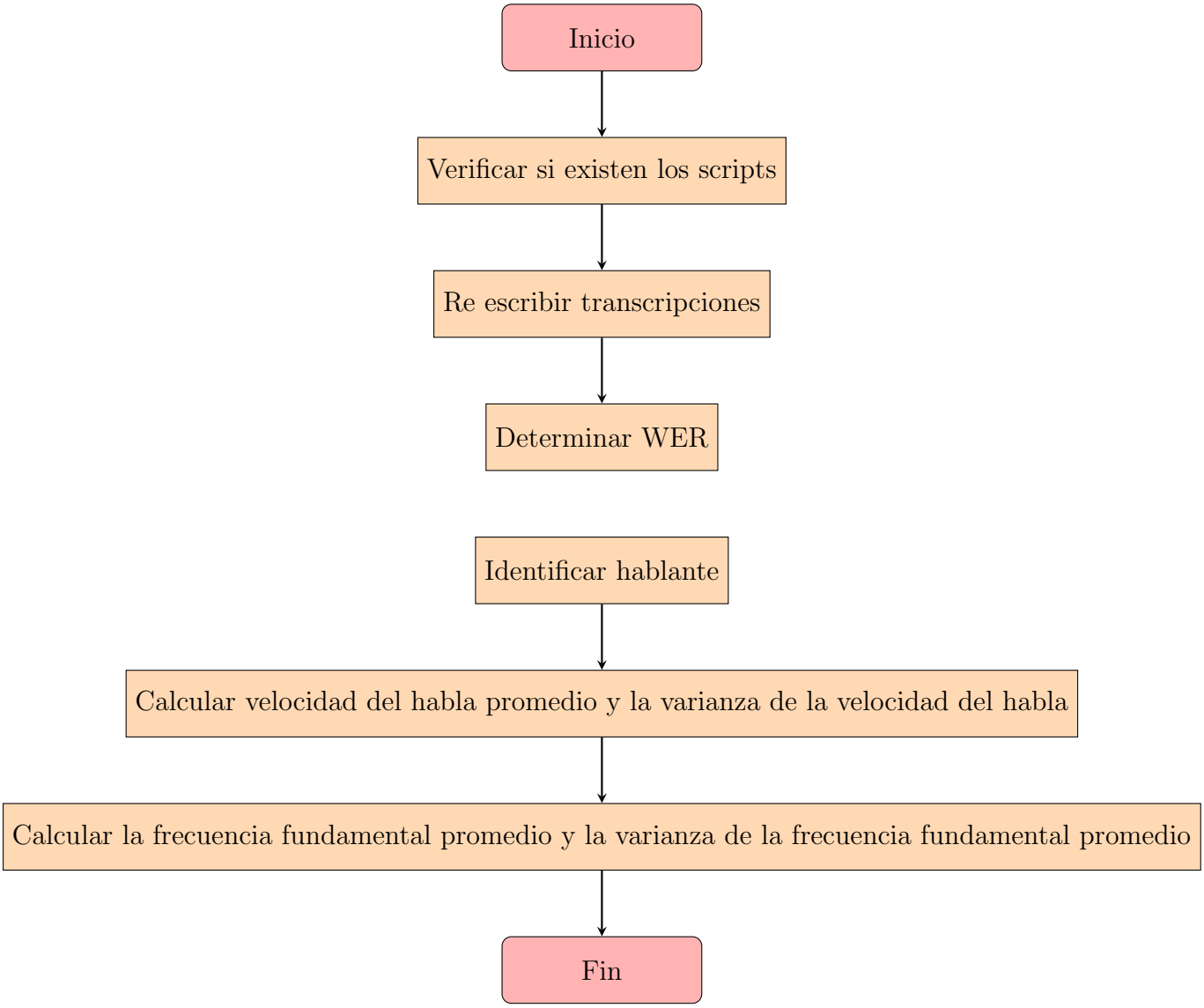
Las funciones dentro de main.py que amplifican, aplican el denoiser y crean las transcripciones tiene como parámetros de entrada los siguientes directorios:

- Amplificador:
  - **Carpeta de entrada:** `/content/Grabaciones_de_lejos`
  - **Carpeta de salida:** `/content/Grabaciones_amplificadas`
- Denoiser:

- **Carpeta de entrada:** `/content/Grabaciones_de_cerca/` y `/content/Grabaciones_amplificadas/`
- **Carpeta de salida:** `/content/Grabaciones_pre_denoiser/`

- Transcripciones:
  - **Carpeta de entrada:** `/content/Grabaciones_pre_denoiser` y `/content/Audio_denoised_ResembleAI_rk4`
  - **Carpeta de salida:** `/content/Transcripciones_sin_denoiser` y `/content/Transcripciones_con_denoiser_ResembleAI_RK4`

3.2. generador_resultados.py



- Preprocesamiento de transcripciones:

  - **Entrada:** `/content/Texto_Referencia.txt`
  - **Salida:** `/content/Texto_Referencia_preprocesado.txt`
  - **Entrada:** `/content/Transcripciones_sin_denoiser`
  - **Salida:** `/content/Transcripciones_sin_denoiser_preprocesadas`
  - **Entrada:** `/content/Transcripciones_con_denoiser_ResembleAI_RK4`
  - **Salida:** `/content/Transcripciones_con_denoiser_ResembleAI_RK4_preprocesadas`

- Medición de SNR:
  - **Entrada sin denoiser:** `/content/Grabaciones_pre_denoiser`
  - **Archivo CSV:** `Parametros_sin_denoiser.csv`
  - **Entrada con denoiser:** `/content/Audio_denoised_ResembleAI_rk4`
  - **Archivo CSV:** `Parametros_con_denoiser_ResembleAI_RK4.csv`

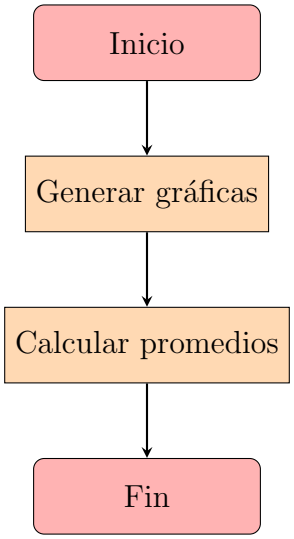
- Determinación de WER:
  - **Entrada sin denoiser:** `/content/Transcripciones_sin_denoiser_preprocesadas`
  - **Salida sin denoiser:** `/content/Resultados_WER_sin_denoiser`
  - **Entrada con denoiser:** `/content/Transcripciones_con_denoiser_ResembleAI_RK4_preprocesadas`
  - **Salida con denoiser:** `/content/Resultados_WER_con_denoiser_ResembleAI_rk4`

- Identificación de hablante:
  - **Referencias:** `/content/Grabaciones_de_cerca/Grabacion1_Punto2_TomadaEnPunto2_Isabel.mp3` y `/content/Grabaciones_de_cerca/Grabacion1_Punto5_TomadaEnPunto5_Coto.mp3`
  - **Carpeta de audios:** `/content/Grabaciones_pre_denoiser` y `/content/Audio_denoised_ResembleAI_rk4`

- Medición de velocidad:
  - **Entrada sin denoiser:** `/content/Transcripciones_sin_denoiser_preprocesadas`
  - **Archivo CSV:** `Parametros_sin_denoiser.csv`

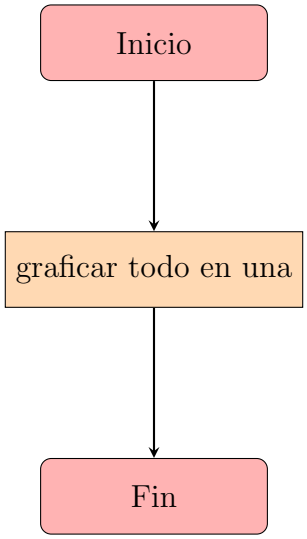
- Cálculo de frecuencia fundamental:
  - **Entrada sin denoiser:** `/content/Grabaciones_de_cerca`
  - **Archivo CSV:** `Parametros_sin_denoiser.csv`

3.3. generador_graficas_y_promedios.py



- **Entradas:**
  - Archivos CSV:
    - `/content/Parametros_sin_denoiser.csv`
    - `/content/Parametros_con_denoiser_ResembleAI_RK4.csv`
- **Salidas:**
  - **Gráficos generados:**
    - **Carpeta para gráficos en PDF:**
      - Ruta: `/content/Resultados_Graficas/Comparacion_Denoisers/PDFs`
      - Archivo: `comparacion_Hablante.pdf`
    - **Carpeta para gráficos en PNG:**
      - Ruta: `/content/Resultados_Graficas/Comparacion_Denoisers/PNGs`
      - Archivo: `comparacion_Hablante.png`
  - **Promedios generados:**
    - `/content/promedios_sin_denoiser.txt`
    - `/content/promedios_denoiser_resemble_ai_rk4.txt`

3.4. generador_graficas_2.py



Entradas:

- **Ruta:** `/content/promedios_sin_denoiser.txt`

Salidas:

- **Carpeta para gráficos en PDF:**
  - Ruta: `/content/Resultados_Graficas/Frecuencia_y_Velocidad/PDFs`
  - Archivo: `f0_y_sr.pdf`
- **Carpeta para gráficos en PNG:**
  - Ruta: `/content/Resultados_Graficas/Frecuencia_y_Velocidad/PNGs`
  - Archivo: `f0_y_sr.png`

4. Guía paso a paso para ejecutar el proyecto

4.1. Paso 1: Importar archivos de github y copiarlos en /content/

Para importar todos los archivos y grabaciones necesarias desde Github se ejecuta el siguiente comando:

```
!git clone https://github.com/aisa0/Reconstruccion_de_senales_de_audio.git
```

Luego, para poder ejecutar los códigos desde Google Colab, es necesario copiar todos los ejecutables desde la carpeta del repositorio */content/Reconstruccion_de_senales_de_audio* en el directorio /content/, para ello se utiliza el siguiente código:

```
import shutil

# Definir las rutas de origen y destino
source_dir = '/content/Reconstruccion_de_senales_de_audio'
destination_dir = '/content/'

# Copiar todo el contenido del directorio origen al directorio destino
shutil.copytree(source_dir, destination_dir, dirs_exist_ok=True)
```

4.2. Paso 2: Instalar de librerías para main.py

4.2.1. Instalación de librosa y soundile

Se utilizan para procesamiento de audio y carga de archivos.

```
!pip install librosa soundfile
```

4.2.2. Instalación de Deepgram SDK y asyncio

Se instala Deepgram SDK para transcripción, asyncio para asincronía.

```
!pip install deepgram-sdk aiohttp nest_asyncio
```

4.2.3. Instalación de pydub y ffmpeg

Para manipulación de audio, conversión entre formatos y soporte para ffmpeg.

```
!pip install pydub ffmpeg
```

4.3. Paso 3: Clonar repositorio e instalar librerías para denoiser con Resemble-AI

Se clona el repositorio de Resemble AI.

```
!git clone https://github.com/resemble-ai/resemble-enhance.git
```

Se instala la librería resemble-enhance.

```
!pip install resemble-enhance --upgrade --pre
```

4.4. Paso 4: Ejecutar main.py

Se ejecuta el main.py, el cual es el encargado de llamar a las funciones e indicar los directorios a los cuales se le aplica la amplificación, el denoiser y la transcripción de los audios. El diagrama de flujo se puede observar en la Sección 3.1.

- Se encarga de amplificar las grabaciones de lejos, aplicar el denoiser a todos los audios y crear las transcripciones respectivas.
- Se puede ejecutar directamente ejecutando la celda activa de Colab o bien con el siguiente código.

  ```
  !python /content/main.py
  ```

- Antes de ejecutarlo es importante agregar su api_key personal en config.py.
- Para crear la api_key lo puede hacer en la siguiente dirección web de Deepgram, luego de registrarse. Api Key Deepgram Link

Al crear las transcripciones, Deepgram da un parámetro de Confiabilidad que se anota en los archivos Parametros_sin_denoiser.csv y Parametros_con_denoiser_ResembleAI_RK4.csv que da el código como parte de su salida.

4.5. Paso 5 (Opcional - Recomendado): Guardar en Drive los resultados de main.py

Este es un paso no obligatorio pero recomendable considerando que el main.py cada vez que se ejecuta tiene una larga duración de al menos 2h (para 28 grabaciones analizadas) y gasta el dinero asociado al usuario de Deepgram.

Para ello se puede utilizar el siguiente código ubicado en la celda de Google Colab siguiente a la ejecución del main.py, es decir, ejecutar el Paso 5 de la libreta en Google Colab.

Considere cambiar el directorio /ProyectoElectrico_v6 por su carpeta personal donde tiene el proyecto en el Drive.

```
import os
import shutil
from google.colab import drive

# Montar Google Drive
drive.mount('/content/drive')

# Funci n para copiar carpetas a Google Drive
def guardar_en_drive():
    drive_base_dir = "/content/drive/MyDrive/ProyectoElectrico_v6"
    # Definir las carpetas a guardar en Google Drive
    carpetas_a_guardar = [
        "/content/Grabaciones_amplificadas",
        "/content/Grabaciones_pre_denoiser",
        "/content/Audio_denoised_ResembleAI_rk4",
        "/content/Transcripciones_sin_denoiser",
        "/content/Transcripciones_con_denoiser_ResembleAI_RK4",
        "/content/Parametros_sin_denoiser.csv",
        "/content/Parametros_con_denoiser_ResembleAI_RK4.csv",
    ]

    # Crear directorios y copiar los archivos
    for carpeta in carpetas_a_guardar:
        if os.path.exists(carpeta):
            nombre_carpeta = os.path.basename(carpeta)
            destino = os.path.join(drive_base_dir, nombre_carpeta)
            try:
                if os.path.isfile(carpeta):
                    shutil.copy(carpeta, destino)
                else:
                    shutil.copytree(carpeta, destino, dirs_exist_ok=True)
                print(f"{nombre_carpeta} guardado en Google Drive.")
            except Exception as e:
                print(f"Error al guardar {nombre_carpeta}: {e}")
        else:
            print(f"La carpeta {carpeta} no existe.")

# Llamar a la funci n
guardar_en_drive()
```

4.6. Paso 6 (Opcional - Recomendado): Importar desde Drive los resultados de main.py

- Importa los archivos generados por el main.py que previamente fueron guardados en Drive.
- Es útil en caso de que se acabe la sesión en Colab tener un punto de retorno y así no tener que volver a correr el main.py.

- Funciona solo si los resultados fueron previamente guardados en Drive.

Para ello se puede usar el siguiente código:

```
import os
import shutil
from google.colab import drive

# Montar Google Drive
drive.mount('/content/drive')

# Funci n para copiar carpetas desde Google Drive a /content/
def copiar_a_content():
    # Definir la ruta base de Google Drive
    drive_base_dir = "/content/drive/MyDrive/ProyectoElectrico_v6"

    # Definir las carpetas a copiar
    carpetas_a_copiar = [
        "Grabaciones_amplificadas",
        "Grabaciones_pre_denoiser",
        "Audio_denoised_ResembleAI_rk4",
        "Transcripciones_sin_denoiser",
        "Transcripciones_con_denoiser_ResembleAI_RK4",
        "Parametros_sin_denoiser.csv",
        "Parametros_con_denoiser_ResembleAI_RK4.csv",
    ]

    # Crear directorios y copiar los archivos
    for carpeta in carpetas_a_copiar:
        fuente = os.path.join(drive_base_dir, carpeta)
        destino = os.path.join("/content", carpeta)

        if os.path.exists(fuente):
            try:
                if os.path.isfile(fuente):
                    shutil.copy(fuente, destino)
                else:
                    shutil.copytree(fuente, destino, dirs_exist_ok=True)
                print(f"{carpeta} copiado a /content/.")
            except Exception as e:
                print(f"Error al copiar {carpeta}: {e}")
        else:
            print(f"La carpeta o archivo {fuente} no existe.")

# Llamar a la funci n
copiar_a_content()
```

4.7. Paso 7: Ejecutar generador_*resultados.py*

4.7.1. Instalar librerías para generador_resultados.py

Instalación de unidecode

Se utiliza para normalización de texto, útil en preprocesamiento de las transcripciones para quitar tildes, cambiar mayúsculas por minúsculas y signos de puntuación.

```
!pip install unidecode
```

Instalación de speechbrain

Se utilizan modelos de deep learning de audio, identificación de hablante.

```
!pip install speechbrain
```

4.7.2. generador_resultados.py

- Reescribe las transcripciones sin signos de puntuación y todo en minúsculas.
- Genera los resultados de WER, Speed Rate y Frecuencia Fundamental y los anota en el .csv correspondiente.
- Muestra en consola el valor del Speed Rate promedio y su varianza.
- Muestra en consola el porcentaje de aciertos para la identificación de hablante.
- Si se desea también puede generar el SNR pero hay que descomentarlo primero.

Para ejecutar este código se puede hacer directamente desde la celda de Google Colab o bien con el siguiente código:

```
!python /content/generador_resultados.py
```

4.8. Paso 8: Ejecutar generar gráficas y archivos con datos promedios

4.8.1. generador_graficas_y_promedios.py

- Este código genera las gráficas del WER y Confiabilidad para ambos hablantes tanto en formato png como pdf y da en un .txt los valores de los promedios del WER, Confiabilidad, Velocidad del habla (y su varianza) y la Frecuencia Fundamental (y su varianza) para cada hablante.
- Se puede ejecutar directamente en la celda de Google Colab o bien con ejecutando la siguiente línea de código:

```
!python /content/generador_graficas_y_promedios.py
```

- Para funcionar requiere los archivos .csv generados con generador_resultados.py.
- Se generan archivos .txt con los valores promedios de cada parámetro estudiado.

4.8.2. generador_graficas_2.py

- Este código genera las gráficas tanto en formato png como pdf de las frecuencias fundamentales y velocidades del habla para cada hablante.
- Se puede ejecutar directamente en la celda de Google Colab o bien con ejecutando la siguiente línea de código:

```
!python /content/generador_graficas_2.py
```

- Para funcionar requiere el archivo /content/promedios_sin_denoiser.txt generado por el generador_graficas_y_promedios.py.

5. Implementación de un sistema basado en deep learning para la reconstrucción de las señales

Esta implementación esta basada en el denoiser de Facebook que se puede encontrar en el repositorio: Facebook Denoiser.

5.1. Paso 9: Corroborar que todas las grabaciones tengan la misma duración

Para poder entrenar el modelo de Facebook con los datos creados a partir de las grabaciones experimentales, se debe asegurar que la duración entre las grabaciones de cerca y su pareja de lejos son de la misma. Para ello se puede ejecutar el código asociado al Paso 9 en la libreta de Google Colab.

5.2. Paso 10: Organizar grabaciones para entrenar denoiser

Luego de corroborar la duración de las grabaciones, es necesario organizar las grabaciones en carpetas para posterior creación de datasets. Para esto se toman las grabaciones que previamente se corroboró o ajustó la duración y se guardan en distintas carpetas /content/Grabaciones_entrenamiento_D_FB/noisy para las grabaciones de lejos y /content/Grabaciones_entrenamiento_D_FB/clean para las grabaciones de cerca, excepto las grabaciones en el punto 5 que se utilizan para la Validación del modelo por lo que van en la carpeta /content/Validation/noisy y /content/Validation/clean (lejos y cerca respectivamente) y las grabaciones del punto 6 que se utilizan para probar el modelo y van en la carpeta /content/Test/noisy y /content/Test/clean (lejos y cerca respectivamente). El código que se encarga de dicha organización se puede correr directamente de la celda de Google Colab y toma la carpeta /content/Grabaciones_Recortadas, generada en el paso anterior, como entrada.

5.3. Paso 11: Clonar repositorio e instalar librerías para denoiser de Facebook

En esta sección se clona el repositorio de denoiser de Facebook y se instalan las librerías asociadas. Además, se agrega el comando para eliminar la carpeta del denoiser en caso de que se haya creado previamente para evitar conflictos.

```
# Eliminar la carpeta anterior si exist a para evitar conflictos
!rm -rf denoiser

# Clonar el repositorio oficial
!git clone https://github.com/facebookresearch/denoiser.git

# Moverse al directorio y realizar la instalacion
%cd denoiser
!pip install .
!pip install pesq
```

5.4. Paso 12: Generación de los datasets para el entrenamiento, validación y pruebas

5.4.1. Entrenamiento

Se crean los directorios donde estarán los .json generados para el entrenamiento.

```
out = "egs/mydataset/train"
!mkdir -p $out
```

Se crean los .json de entrenamiento.

```
!python -m denoiser.audio /content/Grabaciones_entrenamiento_D_FB/clean/ >
    $out/clean.json
!python -m denoiser.audio /content/Grabaciones_entrenamiento_D_FB/noisy/ >
    $out/noisy.json
```

5.4.2. Validación

Se crean los directorios donde estarán los .json generados para la validación.

```
out = "egs/mydataset/valid"
!mkdir -p $out
```

Se crean los .json de validación.

```
!python -m denoiser.audio /content/Validation/clean/ > $out/clean.json
!python -m denoiser.audio /content/Validation/noisy/ > $out/noisy.json
```

5.4.3. Pruebas

Se crean los directorios donde estarán los .json generados para la validación.

```
out = "egs/mydataset/test"
!mkdir -p $out
```

Se crean los .json de pruebas.

```
!python -m denoiser.audio /content/Test/clean/ > $out/clean.json
!python -m denoiser.audio /content/Test/noisy/ > $out/noisy.json
```

5.5. Paso 13: Cambiar el contenido del debug.yaml

En el archivo debug.yaml ubicado en /content/denoiser/conf/dset/debug.yaml se deben escribir las nuevas rutas donde se encuentran los datasets nuevos de entrenamiento, validación y pruebas para lo que se ejecuta el siguiente código desde la celda de Colab.

```
# Define el contenido del archivo YAML
yaml_content = """
dset:
  train: egs/mydataset/train   # path to train folder, should contain both
      a noisy.json and clean.json file
  valid: egs/mydataset/valid   # path to the validation folder.
                        # If not set, the last epoch is kept rather than
                           the best
  test: egs/mydataset/test    # Path to the test set. Metrics like STOI
     and PESQ are evaluated on it
                        # every `eval_every` epochs.
  noisy_json: egs/mydataset/train/noisy.json  # files to enhance. Those
     will be stored in the experiment
                                        # `samples` folder for easy
                                           subjective evaluation of the
                                           model.
  noisy_dir:
  matching: sort       # how to match noisy and clean files. For Valentini
     , use sort, for DNS, use dns.
eval_every: 2
"""

# Ruta al archivo a sobrescribir
file_path = "/content/denoiser/conf/dset/debug.yaml"

# Sobrescribe el archivo con el contenido proporcionado
with open(file_path, "w") as file:
    file.write(yaml_content)

print(f"El archivo {file_path} ha sido sobrescrito exitosamente.")
```

5.6. Paso 14: Entrenar el modelo de denoiser de Facebook con los datos experimentales

Se entrena el modelo utilizando el siguiente código desde una celda de Google Colab.

```
!python train.py
```

6. Trabajo Futuro

- Analizar la efectividad de los distintos modelos de denoiser de Facebook para el idioma español y, de ser posible, diferentes acentos.
- Analizar la efectividad de los distintos modelos del enhacer de Resemble AI para el idioma español con diferentes acentos.
- Analizar la efectividad de la aplicación de filtros pasa altos como el Butterworth para disminuir el ruido en las señales de audio.
- Comparar los modelos de denoiser de Facebook con los de Resemble AI.
- Investigar el modelo ASR para transcripción de audios.

Referencias

[1] Resemble AI. *Introducing Resemble Enhance: Open Source Speech Super Resolution AI Model*. https://www.resemble.ai. Accessed: 2024-11-27. Dic. de 2023.

[2] Z. Huang. *WER-in-python/wer.py*. https://github.com/zszyellow/WER-in-python/blob/master/wer.py#L9. Accessed: 2024-12-01. 2020.

[3] Deepgram Inc. *Deepgram Automatic Speech Recognition Platform*. https://www.deepgram.com. Accessed: 2024-11-27. 2024.

[4] P. Reddy, S. Dubey y A. Prakash. *Speech Denoising Using Neural Networks*. https://github.com/facebookresearch/denoiser. Accessed: 2024-11-27. 2020.