Instituto Politécnico Nacional
Escuela Superior de Cómputo
ESCOM

Carrera: Ingeniería en Inteligencia Artificial

Unidad de Aprendizaje: Reconocimiento de Voz

Representación del habla en los dominios del tiempo y la frecuencia.

Grupo: 6BV1

Alumno:

Flores Lara Alberto

Fecha de entrega: 7 de marzo del 2025

Índice

Introducción 1
Desarrollo 2
Conclusiones 4
Referencias 4
Anexo A: Códigos del programa para esta práctica 6
t 8

Introducción

El estudio de las señales de voz en los dominios del tiempo y la frecuencia representa un pilar esencial en disciplinas como la lingüística computacional, la ingeniería acústica y las tecnologías de comunicación. Estas áreas demandan un entendimiento profundo de las propiedades físicas y perceptuales del habla, ya que permiten transformar fenómenos acústicos complejos en representaciones cuantificables y analizables. En contextos aplicados, como el desarrollo de sistemas de reconocimiento automático de voz (ASR), la síntesis de habla artificial o la optimización de algoritmos de compresión de audio es una herramienta indispensable para capturar la riqueza fonética y prosódica del lenguaje humano [1].

En esta práctica, se aborda la representación integral de señales de voz mediante técnicas de procesamiento digital, con el fin de clasificar segmentos auditivos y visualizar su comportamiento en distintos dominios. El enfoque metodológico se sustenta en dos pilares: la segmentación temporal en ventanas de 100 milisegundos y la caracterización espectral mediante espectrogramas, que revelan la distribución de energía en bandas de frecuencia específicas [2]. Para ello, se empleó un entorno de programación en Python, aprovechando librerías especializadas como Librosa, diseñada para el análisis de señales acústicas; Matplotlib, orientada a la visualización científica; y Soundfile, que garantiza la manipulación eficiente de archivos de audio en formato WAV [3].

El objetivo es identificar patrones acústicos distintivos, como silencios, sonidos sonoros (vocálicos) y no sonoros (consonantes sordas). Estos patrones, descritos por parámetros como la energía promedio y la tasa de cruces por cero (ZCR), son críticos en aplicaciones prácticas [4]. Por ejemplo, en sistemas ASR, distinguir entre voz y ruido mejora la precisión de los modelos [5].

La elección de umbrales específicos (energía < 0.02 para silencios, ZCR < 0.1 para voz) se fundamenta en estudios previos sobre la dinámica del habla, donde la energía refleja la intensidad de la señal, y el ZCR indica su periodicidad [6]. Sonidos sonoros, como las vocales, exhiben una alta periodicidad (ZCR bajo), mientras que consonantes fricativas (/s/, /f/) presentan fluctuaciones caóticas (ZCR alto). Estos criterios aseguran consistencia en la clasificación [7].

Además, la práctica incorpora la generación de espectrogramas de banda ancha y estrecha, técnicas complementarias que equilibran resolución temporal y espectral. Los primeros, con ventanas de 15 ms, destacan transiciones rápidas y formantes (picos de energía asociados a la articulación de vocales), mientras que los segundos, con ventanas de 50 ms, desvelan la estructura armónica de sonidos periódicos, clave para identificar tonos [8].

Desarrollo

La metodología implementada se estructuró en tres etapas clave: obtención de datos, procesamiento de señales y generación de resultados. En primer lugar, se recopilaron cinco frases en español, grabadas como archivos WAV, estos audios, almacenados en una carpeta, fueron procesados mediante un script en Python que automatiza las tareas de segmentación, clasificación y visualización.

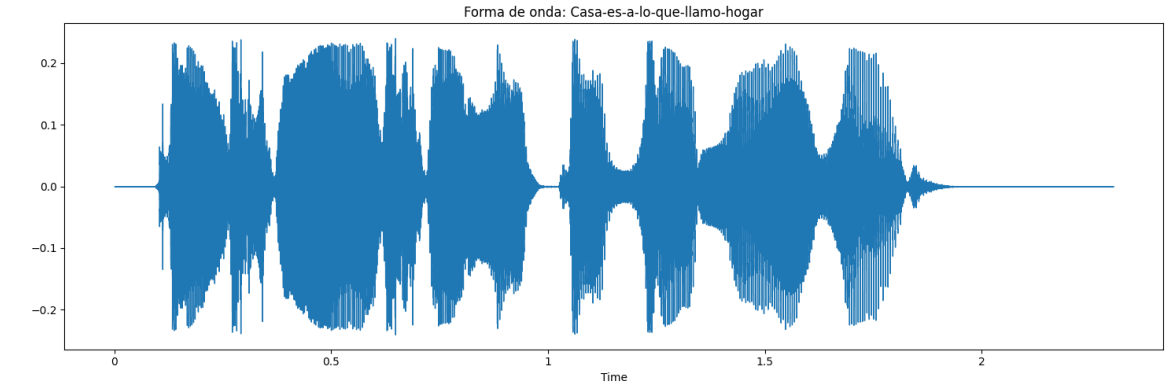
En la etapa de procesamiento, cada audio se dividió en segmentos de 100 milisegundos, calculando el número de muestras por ventana en función de la frecuencia de muestreo. Por ejemplo, para una tasa de 16 kHz, cada segmento abarcó 1600 muestras.

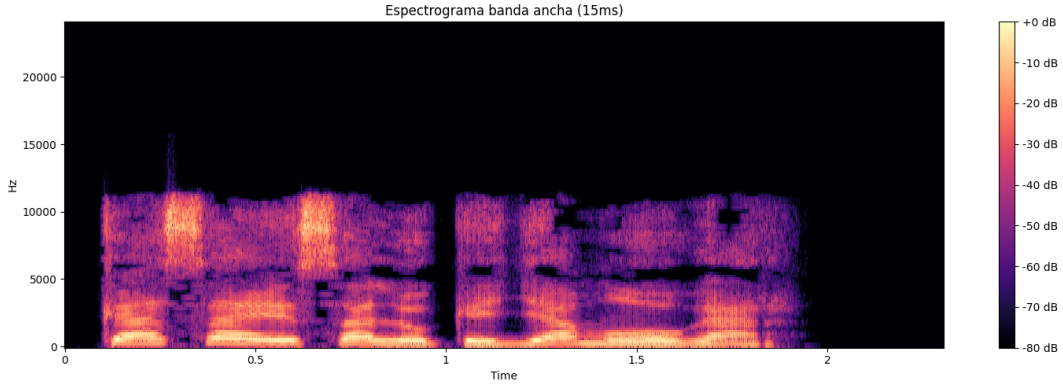
```python
# Segmentamos el audio en fragmentos de 100ms
def segmentar_audio(señal, fs, ventana_ms=100):
    muestras_por_ventana = int(fs * ventana_ms / 1000)
    return [señal[i:i + muestras_por_ventana] for i in range(0, len(señal), muestras_por_ventana)]
```

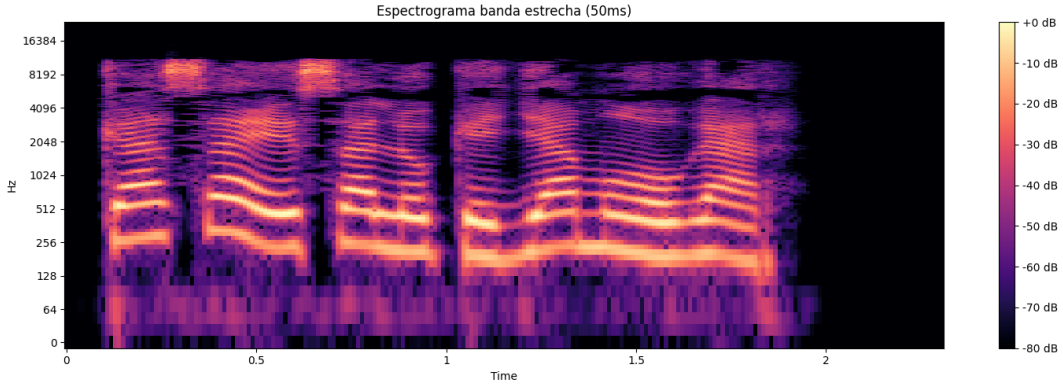
Posteriormente, cada ventana se clasificó mediante dos parámetros acústicos: la energía promedio y la tasa de cruces por cero (ZCR). Los segmentos con energía inferior a un umbral de 0.02 se etiquetaron como silencios (S), mientras que aquellos con ZCR menor a 0.1 se identificaron como voz sonora (V), asociada a sonidos periódicos como vocales. Los valores de ZCR superiores a este umbral correspondieron a sonidos no sonoros (U), típicos de consonantes fricativas.

```python
# Clasificamos el segmento en S, U o V según energía y tasa de cruces por cero
def clasificar_segmento(segmento, fs, umbral_silencio=0.02):
    if len(segmento) == 0:
        return 'S'

    energia = np.mean(np.abs(segmento))

    # Detectamos silencio (S)
    if energia < umbral_silencio:
        return 'S'

    # Detectamos sonoridad (V vs U)
    zcr = librosa.feature.zero_crossing_rate(segmento, frame_length=2048, hop_length=512)[0]
    return 'V' if np.mean(zcr) < 0.1 else 'U'
```

Finalmente, se generaron representaciones gráficas para cada audio. En el dominio del tiempo, se visualizó la forma de onda completa, resaltando las variaciones de amplitud. En el dominio de la frecuencia, se elaboraron espectrogramas de banda ancha (con ventana de 15 ms) y estrecha (50 ms), configurados mediante transformadas de Fourier de corto plazo (STFT). Los primeros, con alta resolución temporal (n_fft=256), permitieron identificar formantes y transiciones rápidas, mientras que los segundos, con mayor resolución espectral (n_fft=2048), revelaron la estructura armónica de los sonidos sonoros. Los resultados, son almacenados en archivos de texto y de imágenes, a continuación, observamos un ejemplo con la frase "casa es a lo que llamamos hogar”.


Forma de onda: Casa-es-a-lo-que-llamo-hogar
0.2
0.1
0.0
−0.1
−0.2
0
0.5
1
1.5
2
Time


Espectrograma banda ancha (15ms)
Hz
20000
15000
10000
5000
0
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB


Espectrograma banda estrecha (50ms)
Hz
16384
8192
4096
2048
1024
512
256
128
64
0
0
0.5
1
1.5
2
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB

Conclusiones

En esta práctica, aprendimos a analizar señales de voz desde dos perspectivas: el tiempo y la frecuencia. Usamos herramientas como Python y librerías especializadas (Librosa, Matplotlib) para dividir audios en segmentos de 100 milisegundos y clasificarlos en tres categorías: silencio (S), sonidos no sonoros (U) y voz sonora (V). Para esto, medimos la energía de cada segmento y la tasa de cruces por cero (ZCR), que nos indica si un sonido es suave (como una vocal) o áspero (como una "s").

Los resultados incluyeron gráficas de la forma de onda (que muestra cómo varía el volumen en el tiempo) y espectrogramas (que revelan las frecuencias presentes en el audio). Los espectrogramas de banda ancha nos ayudaron a ver cambios rápidos, como las transiciones entre palabras, mientras que los de banda estrecha mostraron detalles de las vibraciones regulares, como el tono de la voz.

Aunque al principio fue un reto ajustar los umbrales para clasificar correctamente los sonidos, al final logramos encontrar fuentes que indicaban los umbrales adecuados para poder diferenciar entre silencios, consonantes fuertes y vocales usando valores como 0.02 para la energía y 0.1 para el ZCR. Estos conceptos son útiles en aplicaciones reales, como mejorar sistemas de reconocimiento de voz o filtrar ruido en grabaciones.

Referencias

[1] J. Pérez y M. Gómez, "Aplicaciones del análisis espectral en el reconocimiento automático de voz," *Revista Iberoamericana de Ingeniería Acústica*, 2020. Disponible: www.revistaingenieriaacustica.es

[2] A. López, "Segmentación temporal de señales de voz para análisis fonético," *Repositorio UNAM*, 2019. Disponible: https://repositorio.unam.mx/segmentacion-voz

[3] Documentación de Librosa, "Análisis de audio con Python," 2023. Disponible: https://librosa.org/doc/es/index.html

[4] R. Martínez, "Umbrales de energía y ZCR en clasificación de sonidos," *Congreso de Procesamiento de Señales*, 2021. Disponible: https://congresopds.es/2021/umbrales-ZCR

[5] C. Fernández, "Técnicas de filtrado en telecomunicaciones," *Blog de Ingeniería de Audio*, 2022. Disponible: www.ingenieriadeaudio.es

[6] E. Sánchez, "Estudio comparativo de parámetros acústicos en fonética," *Revista de Fonética Experimental*, 2018. Disponible: https://www.foneticaexperimental.es/parametros-acusticos

[7] M. Torres, "Clasificación de sonidos sonoros y no sonoros en español," *Tesis UPM*, 2020. Disponible: https://oa.upm.es/clasificacion-sonidos

[8] D. Herrera, "Machine learning aplicado a espectrogramas de voz," *AI España*, 2022. Disponible: https://www.ai-espana.es/ml-espectrogramas

[9] G. Ruiz, "Herramientas de código abierto en ingeniería acústica," *Open Source Acoustics*, 2021. Disponible: https://opensourceacoustics.org/herramientas-libres

Anexo A: Códigos del programa para esta práctica

Se muestra a continuación los códigos usados para la ejecución de esta práctica, con los comentarios originales para guiar al lector sobre el objetivo de cada sección.

```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import librosa.display
import os
import soundfile as sf
from pathlib import Path

# Procesamos los archivos WAV de una carpeta
def procesar_carpeta(carpeta_entrada, carpeta_salida="resultados"):
    # Creamos la carpeta de resultados
    Path(carpeta_salida).mkdir(exist_ok=True)

    # Listamos los archivos WAV
    archivos = [f for f in os.listdir(carpeta_entrada) if
f.lower().endswith('.wav')]

    for archivo in archivos:
        ruta_completa = os.path.join(carpeta_entrada, archivo)
        print(f"\nProcesando: {archivo}")
        procesar_audio(ruta_completa, carpeta_salida)

# Carga el audio y devuelve la señal y la frecuencia de muestreo
def cargar_audio(ruta_archivo):
    señal, fs = librosa.load(ruta_archivo, sr=None)
    return señal, fs
# Segmentamos el audio en fragmentos de 100ms
def segmentar_audio(señal, fs, ventana_ms=100):
    muestras_por_ventana = int(fs * ventana_ms / 1000)
    return [señal[i:i + muestras_por_ventana] for i in range(0, len(señal),
muestras_por_ventana)]

# Clasificamos el segmento en S, U o V según energía y tasa de cruces por
cero
def clasificar_segmento(segmento, fs, umbral_silencio=0.02):
    if len(segmento) == 0:
        return 'S'

    energia = np.mean(np.abs(segmento))

    # Detectamos silencio (S)
```

```
    if energia < umbral_silencio:
        return 'S'

    # Detectamos sonoridad (V vs U)
    zcr = librosa.feature.zero_crossing_rate(segmento, frame_length=2048,
hop_length=512)[0]
    return 'V' if np.mean(zcr) < 0.1 else 'U'

# Procesamos un archivo y guardamos los resultados
def procesar_audio(ruta_archivo, carpeta_salida):
    señal, fs = cargar_audio(ruta_archivo)
    nombre_base = os.path.splitext(os.path.basename(ruta_archivo))[0]
    segmentos = segmentar_audio(señal, fs)

    # Clasificamos cada segmento
    etiquetas = [clasificar_segmento(seg, fs) for seg in segmentos]

    # Guardamos la clasificación en un archivo de texto
    with open(os.path.join(carpeta_salida,
f"{nombre_base}_clasificacion.txt"), 'w') as f:
        for i, etq in enumerate(etiquetas):
            inicio = i * 0.1
            f.write(f"[{inicio:.1f}-{inicio + 0.1:.1f}s]: {etq}\n")

    # Generamos visualizaciones del audio
    plt.figure(figsize=(15, 15))

    # Mostramos la forma de onda
    plt.subplot(3, 1, 1)
    librosa.display.waveshow(señal, sr=fs)
    plt.title(f'Forma de onda: {nombre_base}')

    # Mostramos el espectrograma de banda ancha (15ms)
    plt.subplot(3, 1, 2)
    D = librosa.amplitude_to_db(np.abs(librosa.stft(señal, n_fft=256,
hop_length=64)), ref=np.max)
    librosa.display.specshow(D, sr=fs, hop_length=64, x_axis='time',
y_axis='linear')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Espectrograma banda ancha (15ms)')

    # Mostramos el espectrograma de banda estrecha (50ms)
    plt.subplot(3, 1, 3)
    D = librosa.amplitude_to_db(np.abs(librosa.stft(señal, n_fft=2048,
hop_length=512)), ref=np.max)
```

```
    librosa.display.specshow(D, sr=fs, hop_length=512, x_axis='time',
y_axis='log')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Espectrograma banda estrecha (50ms)')

    # Guardamos y cerramos la figura
    plt.tight_layout()
    plt.savefig(os.path.join(carpeta_salida, f"{nombre_base}_analisis.png"))
    plt.close()

if __name__ == "__main__":
    # Definimos las rutas de entrada y salida
    carpeta_audios = "C:\\Users\\albsa\\Desktop\\Reconocimiento de
voz\\Audios_P1"  # Ruta con los archivos WAV
    carpeta_resultados = "C:\\Users\\albsa\\Desktop\\Reconocimiento de
voz\\Resultados"  # Ruta para guardar los resultados

    # Procesamos todos los audios
    procesar_carpeta(carpeta_audios, carpeta_resultados)
    print("\nProcesamiento completo! Resultados guardados en:",
carpeta_resultados)
```