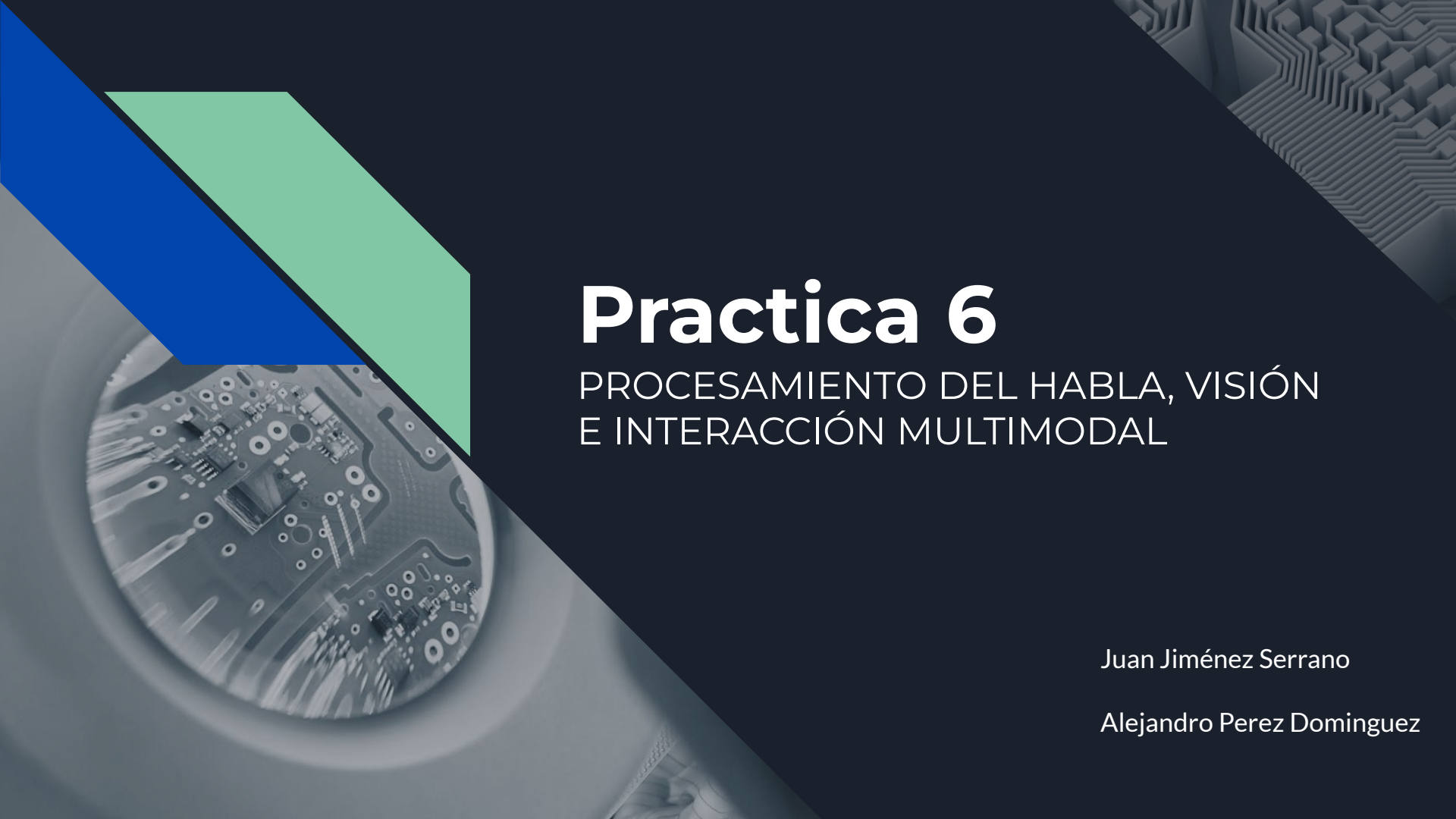


Practica 6

PROCESAMIENTO DEL HABLA, VISIÓN E INTERACCIÓN MULTIMODAL

Juan Jiménez Serrano

Alejandro Perez Dominguez

Índice

Planificación de los difonos del lenguaje

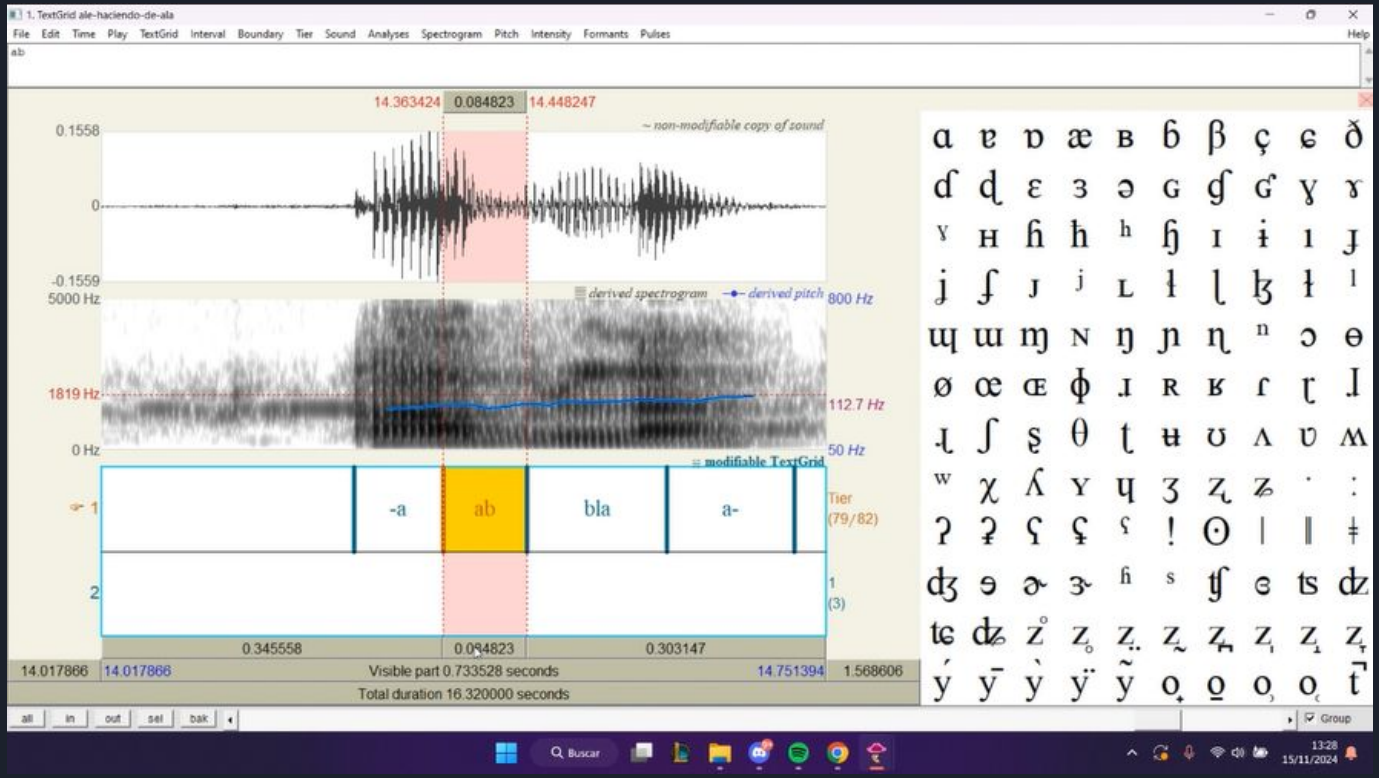
-a	a-			
-b	ba	ab		-bl
-f	fa	af		-fl
-l	la	al		
	ma	am		
-t	ta	at		-tl
-s	sa	as		
	s-	sb		
		sf		
		sl		
		st		

-A	A-	
	bA	Ab
	fA	Af
	lA	Al
	mA	Am
	tA	At
	sA	As

Palabras contenedoras

astas a abas fas alas matasa afalama asba asfas aslas asas flas abla tla

Obtención de los difonos:



Código en Python: (Verificar Cadena)

```
def checkstring(frase: str) -> bool:
    # Verificar si la frase contiene espacios o caracteres
inválidos
    if any(char not in "aAbflmts?" for char in frase):
        return False

    # Verificar si la frase empieza por 'm'
    if frase.startswith('m'):
        return False

    # Verificar si la letra 'm' no está seguida de 's'
    if 'sm' in frase:
        return False

    #Verificar que la letra en la que termina es válida
    if not frase[-1] in ["?", "a", "s", "A"]:
        return False
```

Código en Python: (Verificar Cadena II)

```
# Definir el patrón permitido
    pattern = r"""
    ^(                      # Inicio del patrón
        a?                  # Opcional: Vocal sola (V)
        b|f|t|l|s|m|        # Primera consonante
        bl|fl|tl|           # CCV (b/f/t seguido de l)
        a                   # Vocal (después de C o CC)
        s?                  # Opcional: s para formar VC o CVC
    )+                          # Una o más veces
    \??                     # Opcional: literal '?' al final
    $                           # Fin del patrón
    """

    # Usar expresiones regulares para validar la cadena
    return bool(re.fullmatch(pattern, frase, re.VERBOSE |
re.IGNORECASE))
```

Código en Python: (Diafonización)

```
def diafonizacion(frase: str) -> list:

    # Lista para almacenar los difonos
    difonos = []

    if frase.endswith("?"):
        frase = frase[:-1] #le quita el ultimo caracter
    # Añadir un guion inicial para el primer difono
    difonos.append(f"-{frase[0]}")

    # Generar los difonos intermedios
    for i in range(len(frase) - 1):
        difonos.append(f"{frase[i]}{frase[i + 1]}")
```

Código en Python: (Diafonización II)

```
# Añadir un guion final para el último difono
    difonos.append(f"{frase[-1]}-")

    # Modificar los difonos que contienen una "A" mayúscula y
borrar los fl, tl, bl
    difonos = [
        f"{difono}_acentuado" if 'A' in difono else difono
#si contiene una A se transforma en []_acentuado, si no, se
queda como esta
        for difono in difonos
    ]

    # Devolver la lista de difonos modificados
    return difonos
```

Código en Python: (Crear Audios)

```
def crearaudios(frase: str, output_filename: str):
    # Llamar a CheckString para validar la cadena
    if not checkstring(frase):
        print("La cadena no cumple con las normas.")
    else:
    # Definir la carpeta donde se encuentran los difonos
        difonos_folder = "./Difonos/"

        difonos = diafonizacion(frase)

     # Inicializar un objeto AudioSegment vacío
        audio_output = AudioSegment.silent(duration=100)  # Empezamos con un silencio vacío

    # Definir la duración del crossfade en milisegundos (por ejemplo, 100 ms = 0.1 segundos)
        crossfade_duration = 13
```

Código en Python: (Crear Audios II)

```
    # Iterar sobre los difonos en la lista y concatenarlos
        for difono in difonos:

            difono_filename = difono + ".wav"
            difono_path = os.path.join(difonos_folder,
difono_filename)

        # Comprobar si el archivo existe
            if os.path.exists(difono_path):
            # Cargar el archivo de audio del difono
                audio = AudioSegment.from_file(difono_path,
format="wav")
```

Código en Python: (Crear Audios III)

```
            # Realizar el crossfade y concatenar con el audio
actual
                audio_output = audio_output.append(audio,
crossfade=crossfade_duration)
            else:
                print(f"El archivo {difono_filename} no se
encontró en la carpeta de difonos.")
                break
        audio_output =
audio_output.append(AudioSegment.silent(duration=200))
```

Código en Python: (Crear Audios IV)

```
    # Exportar el audio resultante a un archivo .wav
        audio_output.export("./" + output_filename,
format="wav")
        if frase.endswith("?"):
            transformar_a_pregunta(output_filename,
output_filename)
            print("se ha transformado el audio en una
pregunta")


        print(f"Archivo de audio generado:
{output_filename}")
```

Código en Python: (Transformar a pregunta)

```
def transformar_a_pregunta(input_audio, output_audio):
    # Cargar audio y la tasa de muestreo
    y, sr = librosa.load(input_audio)

    # Duración y punto de división
    duracion = len(y)
    punto_division1 = int(duracion * 0.6)
    punto_division2 = int(duracion * 0.7)

    # Dividir en partes
    inicio = y[:punto_division1]
    medio = y[punto_division1:punto_division2]
    fin = y[punto_division2:]
```

Código en Python:(Main)

```
    # Incrementar el pitch en la parte final
    fin_modificado = librosa.effects.pitch_shift(fin, sr=sr,
n_steps=2)  # +2 semitonos
    medio_modificado = librosa.effects.pitch_shift(medio,
sr=sr, n_steps=0.5)
    inicio_modificado = librosa.effects.pitch_shift(inicio,
sr=sr, n_steps=-0.5)  # +2 semitonos

    # Combinar las partes
    audio_modificado = numpy.concatenate([inicio_modificado,
medio_modificado, fin_modificado])

    # Guardar el archivo modificado
    sf.write(output_audio, audio_modificado, sr)
```

Código en Python:(Main)

```
def main():
    # Crear el parser de argumentos
    parser = argparse.ArgumentParser( description="Crear audios a
partir de una frase." )
    # Definir los parámetros que aceptará el script
    parser.add_argument( "frase", type=str, help="La frase para
generar el audio." )
    parser.add_argument( "output_filename", type=str, help="El
nombre del archivo de salida (ejemplo: salida.wav)" )
    # Parsear los argumentos
    args = parser.parse_args()
    # Llamar a la función CrearAudios con los parámetros
proporcionados
    crearaudios(args.frase, args.output_filename)
if __name__ == "__main__":
    main()

# Llamar al programa: python tts.py astA Salida.wav
```

¿Preguntas?

GitHub