ETSI
ESCUELA TÉCNICA
SUPERIOR DE INGENIERÍA
uhu.es

Procesamiento del habla, visión e interacción multimodal

Practica 6

Juan Jimenez Serrano | Alejandro Perez Dominguez

Introducción

Descripción del proceso de desarrollo

Planificación de los difonos de nuestro lenguaje:

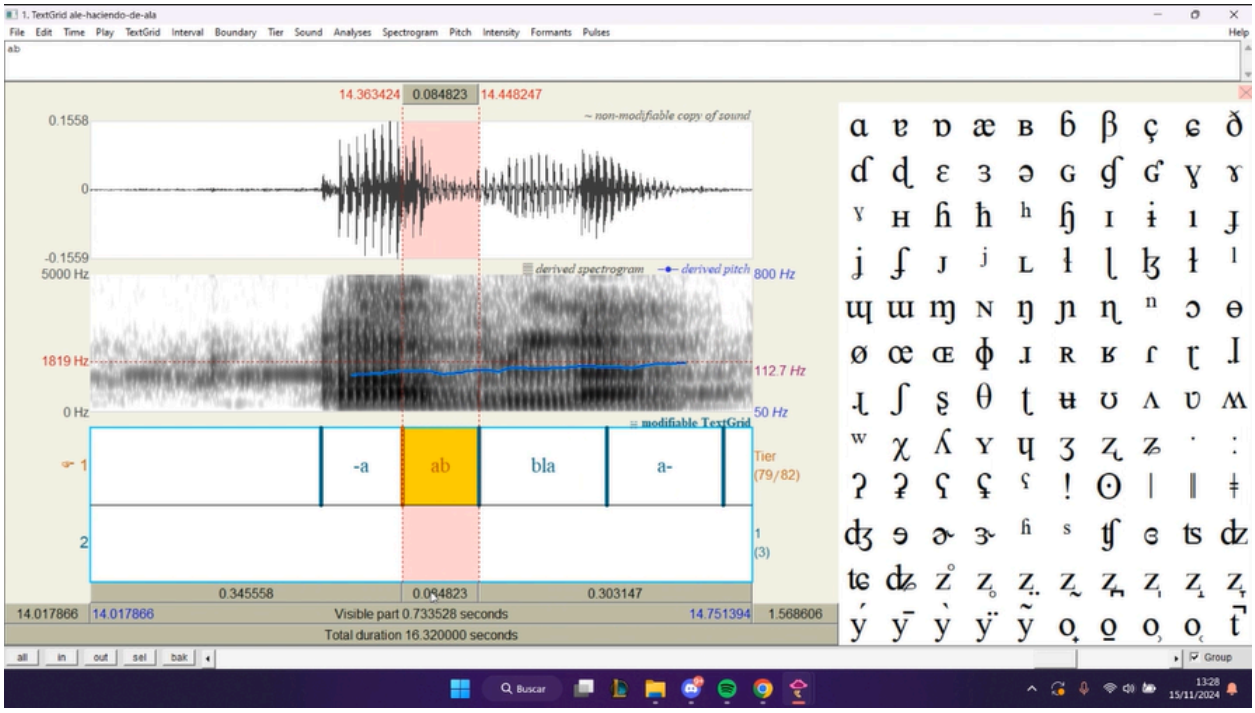
-a	a-			
-b	ba	ab		bl
-f	fa	af		fl
-l	la	al		
	ma	am		
-t	ta	at		tl
-s	sa	as		
	s-	sb		
		sf		
		sl		
		st		
-A	A-			
	bA	Ab		
	fA	Af		
	lA	Al		
	mA	Am		
	tA	At		
	sA	As		

Palabras contenedoras:

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

Obtención de los difonos (sonido):

Una vez grabadas las palabras contenedoras de los difonos, procedemos a capturar los difonos de dichas palabras, para poder generar nuestra biblioteca de difonos.



Programación en Python:

Validación de string de entrada:

Con el método que llamaremos **checkstring** validamos la cadena siguiendo una serie de reglas específicas, especificadas en el enunciado:

1. **Validación de caracteres**:
   - Comprobamos que la cadena contiene únicamente los caracteres válidos **(a, A, b, f, l, m, t, s, ?)**.
2. **Inicio no permitido**:
   - Verificamos que la cadena no comience con el carácter **(m).**
3. **Secuencia inválida**:
   - Nos aseguramos de que la letra **(m)** nunca esté inmediatamente seguida de **(m)**.
4. **Carácter final válido**:
   - Requiere que la cadena termine con uno de los siguientes caracteres: **(?), (a), (s),** o **(A)**.
5. **Validación del patrón**:
   Garantizamos que la cadena sigue una estructura específica:
   - Empieza con una vocal opcional **(a)** seguida de una consonante válida **(b, f, t, l, s, m)** o una combinación consonántica **(bl, fl, tl).**
   - Posteriormente, debe haber la vocal **(a).**
   - Opcionalmente, puede terminar con **(s).**
   - Este grupo puede repetirse una o más veces.
   - Finalmente, la cadena puede terminar con el carácter **(?).**
6. **Resultado final**:
   - Si todas las validaciones anteriores se cumplen, devuelve **True**; de lo contrario, devuelve **False**.

Conversion de string de entrada a array de difonos:

Con el método que llamaremos diafonizacion, procesamos una cadena para generar una lista de difonos siguiendo una serie de reglas específicas, especificadas en el enunciado:

1. **Eliminación del carácter final (?):**
    - Si la cadena termina con el carácter **(?)**, este se elimina para evitar que influya en los difonos.
2. **Generación de difonos inicial y final**:
    - Se añade un difono especial al principio de la lista, formado por - seguido del primer carácter de la cadena (representando el inicio de la secuencia).
    - Se añade un difono especial al final de la lista, formado por el último carácter de la cadena seguido de - (representando el fin de la secuencia).
3. **Generación de difonos intermedios**:
    - Para cada par consecutivo de caracteres en la cadena, se genera un difono (combinación de los dos caracteres adyacentes) y se añade a la lista.
4. **Modificación de difonos con (A) mayúscula:**
    - Los difonos que contienen la letra **(A)** mayúscula se modifican añadiendo el sufijo _acentuado.
5. **Devolución del resultado**:
    - Finalmente, se devuelve una lista con los difonos.

Conversión de array de difonos a archivo de audio:

Con el método que llamaremos **crearaudios**, procesamos la cadena para generar un archivo de audio:

1. **Validación de la cadena**:
    - Llamaremos al método **checkstring**.
2. **Definición de parámetros iniciales**:
    - **Carpeta de los difonos:** Se define la ubicación de los archivos **.wav** correspondientes a los difonos.
    - Lista de difonos: Llamamos al método **diafonizacion** para generar la lista de difonos basada en la cadena.
    - **Audio inicial**: Creamos un segmento de audio vacío (silencio) que servirá como base para concatenar los demás segmentos.
3. **Duración del crossfade**:
    - Utilizaremos la transición (crossfade = 13ms) para suavizar las uniones de los segmentos de audio.
4. **Generación del audio final**:
    - Iteramos sobre cada difono en la lista:
        - Construimos la ruta del archivo .wav correspondiente al difono.
        - Si el archivo existe:
            - Lo cargamos como un segmento de audio.
            - Lo concatenamos con el audio actual, aplicando el crossfade.
        - Si el archivo no se encuentra:
            - Mostramos un mensaje de error y detenemos la generación del audio.
    - Añadimos un breve silencio al final del audio generado para evitar una finalización abrupta.

5. **Transformación en pregunta**:
    - Si la cadena original termina con el carácter **(?)**, llamamos al método **transformar_a_pregunta** para modificar el audio resultante, tratando la prosodia.
6. **Exportación del archivo de audio**:
    - Guardamos el audio generado como un archivo **.wav** (el nombre será la cadena string) en la ubicación especificada por **output_filename**.

Prosodia a la pregunta:

Con el método que llamaremos **transformar_a_pregunta**, procesamos un archivo de audio para modificarlo y adaptarlo a la prosodia interrogativa, siguiendo las siguientes etapas:

1. **Carga del audio**

- Utilizamos **librosa.load** para cargar el archivo de audio desde la ruta **input_audio**. Esto nos proporciona la señal de audio (y) y la tasa de muestreo (sr).

2. **Cálculo de la duración y puntos de división**

- Calculamos la duración total del audio en muestras **(len(y))**.
- Definimos dos puntos de división que dividen el audio en tres partes:
    - **Inicio**: del comienzo hasta el 60% de la duración.
    - **Medio**: del 60% al 70%.
    - **Fin**: del 70% hasta el final.

3. **División del audio en segmentos**

- Separar el audio en las tres partes **(inicio, medio, fin)** según los puntos de división.

4. **Modificación de tonos por segmento**

- **Inicio**: Reducimos ligeramente el tono en 0.5 semitonos para generar un descenso sutil.
- **Medio**: Aumentamos el tono en 0.5 semitonos para crear una transición.
- **Fin**: Incrementamos el tono en 2 semitonos, simulando la elevación característica de las preguntas.

- Usamos **librosa.effects.pitch_shift** para realizar estas modificaciones en los segmentos.

5. **Combinación de los segmentos**

- Concatenamos los segmentos modificados **(inicio_modificado, medio_modificado, fin_modificado)** utilizando **numpy.concatenate** para reconstruir el audio.

6. **Exportación del audio modificado**

- Guardamos el audio resultante en la ubicación especificada por **output_audio** utilizando **soundfile.write (sf.write)**.

Llamada y ejecución del programa:

Para ejecutar el programa deberá llamarse al script desde la terminal **(CMD o powershell)** pasándole dos argumentos, los cuales serán la *frase a generar* y la *ruta del archivo de salida*.

Por ejemplo, estando ya en la carpeta donde se sitúa el script:

```
python tts.py astA Salida.wav
```

Para poder ejecutar el programa sin problemas será necesario tener los archivos de audio de los dífonos en la carpeta “Difonos”la cual deberá estar en la misma carpeta que el Script tts.py

Enlaces:

Repositorio a GitHub:

https://github.com/jujise96/PH_P6

Memoria:

https://docs.google.com/document/d/1Yvgt9VB94zG2fg2jtSy_0NUJ2qEprSqPXWijAeLZbFw/edit?usp=sharing

Presentación:

https://docs.google.com/presentation/d/1pZCmdIoQtN5uvhyOYc9wKYPzv_R6Zi3p9e39txgfjIc/edit?usp=sharing