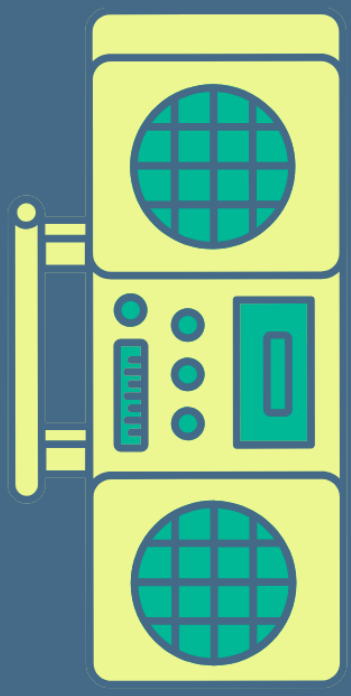
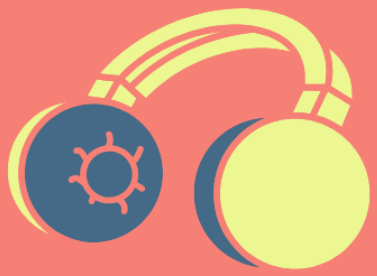
PROCESAMIENTO DEL HABLA, VISIÓN E INTERACCIÓN MULTIMODAL





24
DICIEMBRE

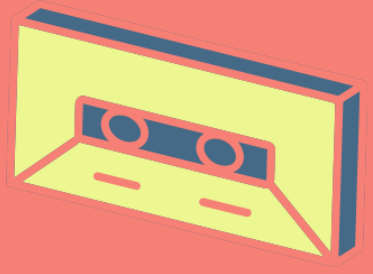
SINTETIZADOR CONCATENATIVO

Por:

Fco Javier Aquino Piosa

José María Corralejos Mora

Ricardo Manuel Ruíz Díaz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA. UHU. 2019-2020

Contenido

Contenido de ilustraciones.

Contenido de tablas.

RESUMEN.

En el presente documento se llevará a cabo una implementación de un sintetizador TTS (Text To Speech), para poder comprender el desarrollo y el posterior éxito de la práctica se documentará desde la creación de nuestra 'base de datos local' de difonos hasta nuestras dificultades en la implementación del código, pasando por cada una de las fases necesarias para la creación de un sistema TTS estudiado en la asignatura Procesamiento del Habla, Visión e Interacción Multimodal.

Capítulo 1: Esquema general.

Para comenzar debemos explicar cómo funciona a groso modo nuestro programa. Tras ser lanzado desde consola deberá recibir una cadena de texto en la que aparezcan los difonos guardados en nuestra base de datos local, una carpeta.

Para ello, en primer lugar, se deberá acomodar cada uno de los difonos que emplearemos (como se explicará en un apartado posterior). Seguidamente, la cadena de texto pasada por parámetro se divide en difonos y se busca cada uno de ellos en el directorio de elementos posibles, en caso de que un difono introducido no se corresponda con ninguna instancia de la carpeta local que contiene todos los difonos necesarios para esta práctica, se lanzará una excepción.

Si la excepción no ha sido lanzada, nuestro programa a llamar a Praat para que se monten todos los archivos .wav en un solo archivo, según los parámetros especificados por el profesor de la asignatura en el enunciado de esta actividad. Ese resultado, es el que ejecutará nuestro programa para cumplir el objetivo principal de esta práctica: ejecutar un sintetizador partiendo de un texto dado.

A continuación, se muestra un esquema ilustrativo en el que se afianzan los pasos que deben de seguir nuestro programa:

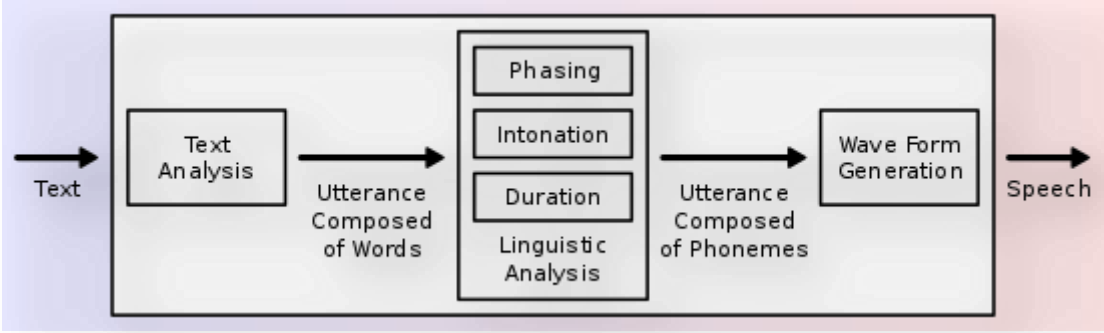


Ilustración 1. Pasos a seguir en un sistema TTS.

Como se puede observar en la imagen y, como en las siguientes páginas demostraremos, se pueden diferenciar tres grandes pasos en un sistema TTS:

- Análisis textual.
- Análisis lingüístico.
- Generación de la onda.

Cada uno de los pasos son totalmente dependendientes del anterior y si estos no se realizan con cautela, el resultado que se obtendrá no será el deseado, pudiéndose complicar el entendimiento de las salidas sonoras.

Capítulo 2: Elaboración de la biblioteca de difonos.

Nuestro sistema TTS se basa en la síntesis concatenativa de difonos, esto es, la creación del sonido final depende del sonido de cada uno de los difonos concatenados, por ejemplo:

Casa → -C + ca + as + sa + a-

Lógicamente, para esta instancia, debemos tener un conjunto de sonidos que abarque (como mínimo) cada uno de los difonos usados en la palabra casa.

Del mismo modo, para nuestro proyecto, hemos tenido que generar un diccionario de difonos. Para simplificar dificultades, se ha debido introducir una variante de acento tonal en la única vocal de nuestro sistema ([e]). Aquí se listan los elementos contenidos en nuestro diccionario particular:

- -s/se/sE/s-
- -t/te/tE/tr/t-
- -f/fe/fE/fr/f-
- -k/ke/kE/kr/k-
- -m/me/mE/m-
- re/rE
- Et/et
- Ek/ek
- Ef/ef
- Em/em
- Er/er
- Es/es
- Kre/kre
- Fre/fre
- Tre/TrE

¿Cuáles han sido los pasos para conseguir dicho diccionario?

Para poder conseguir el diccionario formado por los difonos oportunos, se ha hecho necesario el uso de Amazon Polly, un sintetizador de la compañía estadounidense característico por su naturalidad del habla.

Para grabarlos, se ha usado distintas oraciones [1]en las que una palabra contenedora del difono variaba de posición en las mismas, así como el difono cambiaba de posición respecto a la palabra en multitud de ocasiones. De este modo, conseguimos tener valores correctos, en los que la prosodia original no altera el sonido del difono.

La aplicación on-line permite la descarga de los sonidos, por ende, no se ha hecho efectiva la utilización del software Camtansia Studio 9 como se preveía en un inicio.

Tras descargar los archivos, se ha usado Praat para los recortes **manuales** de estos. Obteniéndose los elementos necesarios para la realización de la práctica.

[1] Mañana iré a **tefekE** un poco → -t/te/ef/fe/ek/kE/E-
Mañana iré a **meresE** un poco → -m/me/er/re/es/sE/E-
Mañana iré a **sefEte** un poco → -s/se/ef/fE/Et/te/e-
Mañana iré a **emEkre** un poco → -e/em/mE/Ek/kr/re/e-
Del mismo modo, se ha seguido este modelo para las palabras: Efre, feketEs,ketrEm,krEref,frEk,trEt.

Capítulo 3: Código del sintetizador.

Para poder generar el sistema, se planteó la posibilidad de programarlo en lenguaje Java o Python. Como bien es sabido, el segundo lenguaje tiene una ventaja destacable sobre Java, su facilidad de implementación siendo capaz de reducir la dificultad de resolución de un problema de manera exponencial. Por contraposición, Python es bastante más lento que Java, pero dado que en este sistema no prima la velocidad o no es una propiedad prioritaria, este grupo se ha decantado por el uso del lenguaje Python, concretamente en su versión 3.7.2.[2]

Una vez decidido el lenguaje de programación que usaremos, comenzamos con la explicación del propio código.

Para comenzar, debemos importar librerías como:

- **Librería OS:** Para poder gestionar y crear (así como eliminar) directorios y realizar llamadas al sistema.
- **Librería shutil:** Para poder eliminar el interior de una carpeta.

Seguidamente, definiremos una función llamada *createfolder(folder)* la cual recibe por parámetro el nombre de una carpeta, procediendo a la comprobación de su existencia. En caso de existir, se elimina la información contenida en ella para evitar la acumulación de residuos que puedan alterar el normal desarrollo de la ejecución del programa, tras la eliminación se crea de nuevo la carpeta. Por otro lado, si no existe se crea.

```
import os
import sutil

def createFolder(folder):
    if os.path.exists(folder):
       print("La carpeta ya existe y se procederá al borrado del interior")
       shutil.rmtree(folder)
       os.system('cls')
       print("Carpeta vaciada correctamente")
       os.mkdir(folder)
    else:
        os.mkdir(folder)
        os.system('cls')
        print("Carpeta creada correctamente")
```

Una vez limpiado el directorio, continuamos con la codificación de la función principal. Es en esta parte del programa donde se pueden diferenciar cada una de las fases del sistema TTS del que se habló en el capítulo anterior.

```
questions = False
folder = input("Escriba el nombre de la carpeta donde se encontrara el
resultado: ")
createFolder(folder)
text = input("\nInserte el texto a reproducir : ")
questions = "?" in text
```

[2] Información y descarga: https://www.python.org/downloads/release/python-372/

Como puede verse, en primer lugar, se declara una variable *bool* a false que nos ayudará a diferenciar si se ha de tratar la prosodia del texto indicado o no. El valor de esta variable cambiará a true dependiendo si se encuentra el signo de interrogación al final de la cadena introducida.

Tras esto, se genera una carpeta que contendrá el resultado de la ejecución del programa, haciendo uso del primer método definido en nuestro código.

Si se ha cambiado el valor de la variable booleana a true, procedemos a hacer un split, quedándonos con la primera parte de la frase, para trabajar con ella.

```
if questions:
    text2 = text.split("?")
    text = text2[0]
```

```
textfile = str(text + ".wav")
text = "-" + text + "-"
```

Seguidamente, declaramos el audio de salida en formato .wav. Se hará uso también de una variable text a la cual le añadimos guiones al inicio y final para poder controlar los difonos con una sola letra.

Una vez creada las variables oportunas, se procederá a ejecutar la mayor parte del código, en la que nos centraremos en la manipulación del texto y audio respectivamente.

Para el correcto funcionamiento, debemos comenzar estudiando si existen las carpetas oportunas, en caso negativo, se debe mostrar un mensaje de error, en nuestro caso mostrará:

"No podemos seguir con el programa por no localizar la carpeta con los Difonos."

En caso afirmativo, debemos seguir con la ejecución del programa donde se realizan las siguientes operaciones:

- Creación de una tupla **t** que contendrá la cadena a buscar.
- Creación de una variable entera **iteraciones** que contiene el tamaño de la cadena introducida.
- Creación de una variable booleana **encontrado** que ayudará a encontrar si hay una consonante seguida de una "-r".

Como bien se aprecia en el enunciado, una 'complicación' surge cuando una consonante va seguida de otra, es por lo que, en esos casos se debe prestar un estudio especial.

Debemos ir recorriendo la cadena de texto introducida para que, si se cumple la condición, cambiar el valor de la variable booleana descrita anteriormente.

Si se dan las condiciones oportunas, debemos tomar la variable pasada por parámetro, cambiamos el valor de la variable booleana por si sucede de nuevo el caso especial posteriormente. Si no se dan las condiciones oportunas, procederemos a concatenar los difonos.

Así mismo, se crea un directorio temporal que almacenará los difonos estudiados, así como un directorio destino y final (posteriormente se explicará su creación).

A modo de código, se muestra lo anteriormente explicado:

```
if os.path.exists("Difonos") & os.path.exists("Scripts_Praat"):
      t = []
      encontrado =  False
      iteraciones = len(text) - 1
      index = 0
      while index < iteraciones:
          if "k" == text[index] or "f" == text[index]  or "t" ==
text[index]:
              if text[index + 1] == "r":
                  encontrado = True

          if encontrado and ((index + 2) < iteraciones):
              t = t + [(text[index] + text[index + 1] + text[index + 2])]
              encontrado = False
              index += 1
          else:
              t = t + [(text[index] + text[index + 1])]

          index += 1

      createFolder("Difonos_temporal")
      destino = os.getcwd() + "\\" + "Difonos_temporal"
      origen = os.getcwd() + "\\" + "Difonos"

      praat = ""
      praat2 = ""
```

Si quisiésemos mostrar nuestra cadena por pantalla para tener en cuenta los difonos a seleccionar podemos implementar:

```
print(q)
len(t)
t = q
```

Una vez contenidos en la variable t los difonos, procedemos a estudiarlos individualmente. En nuestro lenguaje, solo la vocal -e puede tener acentuación y eso lo representamos por -E (-e mayúscula en la cadena introducida). Debemos asegurarnos de que solo la -e puede ser escrita en mayúscula o minúscula, transformando a minúsculas el resto de los difonos. En la siguiente página se muestra un ejemplo contenido en una tabla.

Cadena inicial	Cadena a procesar
MeKEtreFE	mekEtrefE
eSE	eSE
Mereces	mereces

Tabla 1.Procesado entrada.

Como puede observarse, se produce una conversión en las cadenas, conservándose la entonación de la vocal.

Si no se encuentra el difono deseado, significa que no se ha grabado e introducido previamente a nuestro diccionario y, por ende, debe mostrarse un mensaje de error. Este mensaje también veta el uso de la consonante -r al inicio de la palabra, pues como no se ha grabado ese dífono, directamente se lanzaría la excepción.

Una vez realizada la conversión, se procede a generar una llamada al software Praat para poder seleccionar los difonos oportunos del diccionario e ir concatenándolos.

¿Por qué se ha usado la variable i?

Si se concatenase con el mismo nombre, Praat sobreescribiría los sonidos que se llamasen igual. Así, una palabra, por ejemplo “meresetrese” sonaría “meretrese” ya que redunda el difono -se en la palabra.

```
      for i in range(0,len(t)):
            difono = t[i]
            print(difono)

            try:
                  nombre_archivo =  difono + ".wav"
                  if "E" in difono:
                   ubicacion_archivo = origen + "\\" +
nombre_archivo
                  else:
                   difono.lower() #Para ponerlo en minúscula
                   ubicacion_archivo = origen + "\\" +
nombre_archivo

                  shutil.copyfile(ubicacion_archivo, destino + "\\"
+ "%s.wav" %i)

            except:
                  raise Exception("Archivo no encontrado")

            #Script de PRAAT
            archivo = destino + "\\" + "%s.wav" %i
            praat += "Read from file... %s\n" % archivo

            if i == 0:
                  praat2 += "select Sound %s\n" % (i)
            else:
                  o = i + 1
                  praat2 += "plus Sound %s\n" % (i)
```

Una vez unidos, debemos hacer otra llamada a Praat para, ahora sí llevar a cabo la concatenación teniendo en cuenta las distintas características con las que se nos alienta para la elaboración de esta práctica en el enunciado. Una vez obtenido el fichero correspondiente, debemos guardarlo en un archivo correspondiente *".praat"* pudiéndose realizar la llamada a Praat para la ejecución del script.

```
        praat3 = "Concatenate with overlap... 0.0005\n"
        praat3 += "Resample... 16000 16\n"
        praat3 += "Write to WAV file... %s" % textfile
        script_praat = "%s\n%s\n%s" % (praat, praat2, praat3)

        nombre = textfile.split(".")
        archivo_praat = destino + "\\" + nombre[0] + ".praat"

        f = open(archivo_praat,"w")
        f.write(script_praat)
        f.close()

os.system("Praat.exe " + ".\\Difonos_temporal" + "\\" + nombre[0]
+ ".praat")
```

En el siguiente fragmento de código, se procede a guardar la salida de Praat en el directorio resultado.

```
        origen_wav = destino + "\\" + textfile
        directorio_resultado = os.getcwd() + "\\" +  folder + "\\" +
textfile
        shutil.copyfile(origen_wav,directorio_resultado)
```

Finalmente, acabaremos el programa principal verificando si se ha introducido (o no) el signo de interrogación, pudiéndose cambiar la prosodia si es necesario.

```
if questions:
      modificarProsodia(textfile)
else:
      nombre = textfile.split(".")
      os.system("Praat.exe " + ".\\Scripts_Praat\\extraer-pitch-
track.praat " + "..\\Resultado\\" + textfile + " ..\\Resultado\\" +
nombre[0] + ".PitchTier 50 400")
```

En el siguiente apartado, se hará especial atención al cambio de la prosodia.

Capítulo 4: Modificando la prosodia.

Como bien se estudió en clase, hay varias opciones para modificar la prosodia del habla, siendo las posibilidades para cambiar:

- Frecuencia fundamental.
- Duración.
- Intensidad.

En esta práctica nos hemos centrado en los dos primeros parámetros del listado. Para ello, se ha debido modificar el script *reemplazar-pitch-track.praat* añadiendo ciertas líneas de comando para modificar también la duración.

```
[…]
select Sound myfile
dur = Get duration
Create DurationTier: "amplia", 0, dur
Add point: 0.80*dur, 1.2
Add point: dur, 1.2
[…]
select PitchTier myfile
To Pitch... 0.02 'min_pitch' 'max_pitch'
[…]
select Sound myfile
plus Pitch myfile
To Manipulation
select Manipulation myfile
plus DurationTier amplia
Replace duration tier
```

Nota: se han omitido ciertos comandos del script original para hacer más amena la explicación.

Entendamos el script. Inicialmente tomamos la duración del audio a modificar y creamos un `DurationTier [nombre], [inicio], [fin].`

Los parámetros inicio y fin no hacen referencia al audio, si no al campo que se quiere modificar dentro del mismo audio.

Seguidamente añadimos puntos de siguiente modo*; Add point: [lugar] [elevación]*

En nuestro primer caso, por ejemplo, se ha añadido un punto en el tiempo t que corresponde al 80% del tiempo total dur, aumento la duración en un 20%.

Finalmente, se hace una llamada a la opción To Manipulation de Praat donde se añadirían los puntos deseados (*plus DurationTier [nombre_DurationTIer]*)

Una vez generado el script para nuestro propósito, debemos definir una función que haga uso del mismo en Python. Dicha función, recibirá por parámetro el archivo al cual queremos aplicarle la nueva prosodia.

Seguidamente, debemos leer el *.PitchTier* del audio, nos posicionamos en la línea oportuna y hacemos un Split, para tener en cuenta cuántos puntos contamos en el audio.

Tras el Split, podemos determinar qué porcentajes del número total de puntos queremos modificar. Sabemos que, en una oración interrogativa en español, se procede a subir al inicio la frecuencia fundamental, así como al final.

Se ha decidido aumentar dicha frecuencia en el 20% inicial y en el 30% final del audio.

Posteriormente, se recorren esos porcentajes aumentando el valor de la frecuencia fundamental de manera exponencial.

```
def modificarProsodia(textfile):

      nombre = textfile.split(".")
      os.system("Praat.exe " + ".\\Scripts_Praat\\extraer-pitch-track.praat " +
                  "..\\Resultado\\" + textfile + " ..\\Resultado\\" +
nombre[0] + ".PitchTier 50 400")

      file = open(".\\Resultado\\" + nombre[0] + ".PitchTier" , "r")

      linea = []

      for line in file:
            linea = linea + [(line)]

      lineaPuntos = linea[5].split(" = ")
      numPuntos = lineaPuntos[1]

      inicio1 = 0
      fin1 = math.floor(float(numPuntos) * 0.2)
      inicio2 = math.floor(float(numPuntos) * 0.7)
      fin2 = int(numPuntos)

      for i in range(0,fin1):
            l = 8 + (3 * (i))
            dato = linea[l].split(" = ")
            valor = dato[1]
            valorfinal = float(valor) + (5 * (i + 1))
            linea[l] = dato[0] + " = " + str(valorfinal) + "\n"

      for i in range(inicio2,fin2):
            l = 8 + (3 * (i))
            dato = linea[l].split(" = ")
            valor = dato[1]
            valorfinal = float(valor) + (11 * (i + 1))
            linea[l] = dato[0] + " = " + str(valorfinal) + "\n"
```

Se reescribe el archivo *.PitchTier* y se realiza una llamada al sistema a Praat para hacer uso del script *reemplazar-pitch-track.praat* consiguiendo así nuestro propósito final.

```
praat = ""
      for i in range(0,len(linea)):
            praat += linea[i]

      nombre = textfile.split(".")

      f = open(".\\Resultado\\" + nombre[0] +
"_Modificado.PitchTier","w")
      f.write(praat)
      f.close()

      os.system("Praat.exe" + " .\\Scripts_Praat\\reemplazar-pitch-
track.praat" + " ..\\Resultado\\" + textfile + " ..\\Resultado\\" +
nombre[0] + "_Modificado.PitchTier" + " ..\\Resultado\\" + nombre[0]
+ "_Modificado.wav 50 400")
```

Veamos la diferencia de la prosodia entre un archivo en una oración enunciativa (meresetrese) y una oración interrogativa (meresetrese?).

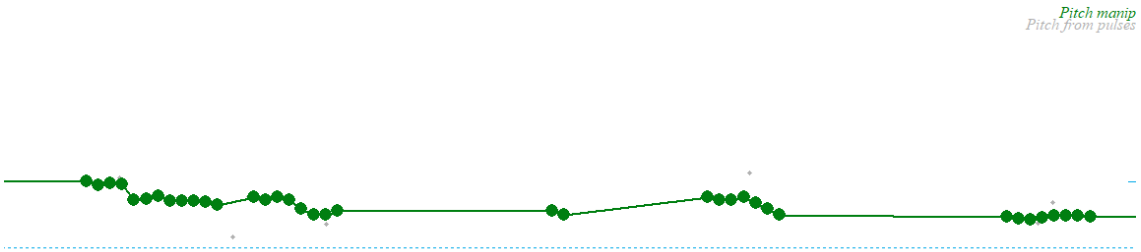


Ilustración 2. Pitch de una oración enunciativa.

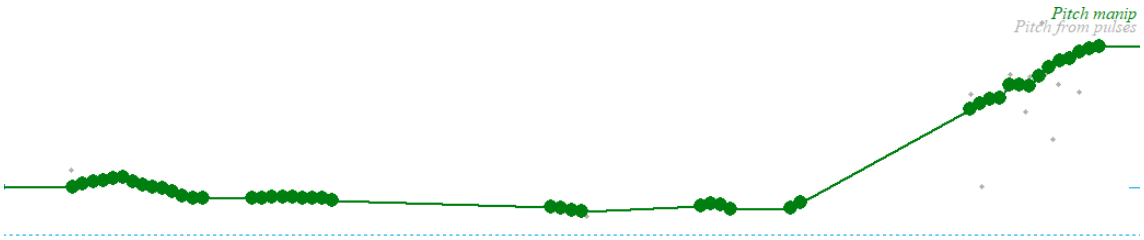


Ilustración 3. Pitch de una oración interrogativa.

Capítulo 5: Complicaciones.

En este apartado, se mostrará a modo de listado algunas de las dificultades surgidas durante el proyecto:

- **Grabación de difonos:** los participantes del grupo no sabíamos qué hacer, teníamos dos opciones, grabarnos nosotros mismos o usar un sintetizador. Nos decantamos por el sintetizador AmazonPolly.
- **Recortes de difonos:** tras muchos recortes, comprobaciones y simulaciones en nuestro programa, obtuvimos los que creemos que son los mejores difonos para el caso que nos ocupa.
- **"Palabras impares":** inicialmente, dado que hay algunos difonos formados por tres letras, cuando la palabra introducida por teclado no tenía un número par de elementos, fallaba.
- **Cambio de duración:** la búsqueda de información y creación del script donde se modificaba la duración fallaba al ejecutarlo, tras varias modificaciones, conseguimos el propósito.

Aunque surgieron algunas complicaciones, se fueron solventando una tras otra hasta conseguir el resultado final.

Capítulo 6: Ejecución.

A continuación, se le indica los pasos que ha de seguir para la correcta ejecución del software desarrollado en esta práctica.

1º) Abra la línea de comandos.

2º) Posiciónese en la carpeta de la práctica que contiene el programa ConcatenadorDifonos.py

```
Microsoft Windows [Versión 10.0.18362.535]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Franc>cd C:\Users\Franc\Desktop\Entregable (3)\Entregable
```

3º) Ejecute el programa del siguiente modo:

```
C:\Users\Franc\Desktop\Entregable (3)\Entregable>python ConcatenaDifonos.py
```

4º) Tras esto, se le pedirá que introduzca la palabra a sintetizar obteniendo la salida en la carpeta "*Resultados"* dentro del directorio principal de la práctica.

Enlaces de interés.

Prácticas no puntuables PVHIM curso 2019-2020.

Amazón AWS:

https://aws.amazon.com/es/

Guía Python:

http://docs.python.org.ar/tutorial/3/index.html

Scripting Praat:

http://phonetics.linguistics.ucla.edu/facilities/acoustic/praat.html

http://stel3.ub.edu/labfon/en/praat-scripts