INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO, ITAM

Laboratorio de Procesamiento Digital de Señales

Práctica Final

Sistema ASR

Nombre del equipo: Fantasy Team

Integrantes del equipo:

- Rubén Romero Ortega, 174178
- Manuel Fernández Verda, 166496

Introducción

El término formante se refiere a los picos en el espectro de la frecuencia de un sonido complejo. Las formantes del sonido de la voz humana son particularmente importantes porque son componentes esenciales en la inteligibilidad del habla. Permiten distinguir los sonidos vocales con sus tres primeras frecuencias formantes. Estas frecuencias de las formantes vocales se determinan en el proceso de articulación del cuerpo humano: la mandíbula, el tracto vocal, el labio, lengua, etc. [1]

Ahora, en procesamiento digital de señales, específicamente del audio, comúnmente se utiliza la la transformada rápida de Fourier. Esta nos sirve para calcular el dominio de la frecuencia en los intervalos de tiempo que se presentan en el espectrograma. La transformada de Fourier tiene un algoritmo inventado por Carl Friedrich Gauss, y es un algoritmo eficiente de la transformada de Fourier, el orden de la FFT es O(NlogN), elimina gran parte de los cálculos repetitivos de la FT [2].

Esta herramienta de la FFT es utilizada por los espectrogramas. Un espectrograma es una representación visual que muestra la evolución temporal del espectro de una señal y sus magnitudes. Su gran ventaja es que nos permite detectar señales de forma más exacta. El espectrograma se construye tomando un trozo de la señal y aplicando una transformada para obtener la imagen de su espectro. Cuando las señales se vuelven muy complicadas el espectrograma no presenta problemas en calcularse, en términos de un Automatic Speech Recognition el uso del espectrograma es crucial, pues permite separar la señal de la palabra en señales más pequeñas aproximándose a un fonema. [3]

Automatic Speech Recognition (ASR) es el proceso y la tecnología relacionada para convertir una señal de voz en su correspondiente secuencia de palabras usando algoritmos implementados en algún dispositivo [4].

El primer dispositivo ASR se usó en 1962 para reconocer los dígitos que decía un usuario dado. Hoy en día, esta tecnología se encuentra en muchas industrias de salud, militares, de telecomunicaciones y en dispositivos personales. [5] . La principal rama de desarrollo de la

tecnología ASR es en las búsquedas por voz, esta búsqueda se está mejorando y cada vez se usa más, en 2018 se registró un promedio de 1000 millones de búsquedas por mes.

Las personas buscan las soluciones más cómodas para realizar sus actividades cotidianas y cada vez desean tener menos contacto físico con las máquinas. Gartner, compañía líder de investigación en el mundo, estimó que en el 2020 se tendrán dispositivos de voz en al menos 75% de los hogares norteamericanos [6].

En un futuro muy cercano, esta tecnología se podría aplicar en dispositivos en los que resulta impráctico el contacto con la computadora. El desarrollo de los ASR disminuye el tiempo de respuesta y el número de personas necesarias para muchos procesos tecnológicos [7].

Un dispositivo ASR funciona con un convertidor analógico a digital para registrar la señal de voz, y una base de datos lo suficientemente grande para realizar modelos de pronunciación, acústicos y estadísticos para comparar la información de entrada y poder decodificarla en la palabra [8]. Para lograr un buen análisis es necesario seguir las reglas lingüísticas del lenguaje que se está decodificando, aplicar un modelo oculto eficiente de cadenas de Markov para determinar con argumentos estadísticos el comportamiento de la señal y utilizar una codificación predictiva lineal para determinar las formantes de la palabra [9].

Los dispositivos necesitan convertir la información analógica en digital para analizar el comportamiento de la señal; sin embargo, esta conversión causa pérdida de información, pues no se puede recuperar de manera exacta la información tras una conversión, ya que no se tiene una memoria infinita, y esto provoca que a veces la señal no se interprete de manera correcta. Esto se ha solucionado usando una memoria más amplia para recibir la señal convertida a digital y lograr una aproximación más exacta.

También existen conflictos en la forma en que hablan los humanos, no todos tienen la misma rapidez, ni el mismo acento, esto provoca que la señal se malinterprete muchas veces al cambiar la forma en la que habla el usuario.

Las aplicaciones de esta tecnología son inmensas, ya que pueden aumentar la productividad y efectividad en muchos campos laborales. Sin embargo, este futuro podría no ser tan cercano si no se siguen mejorando las estrategias y algoritmos para corregir los problemas que se tienen en esta tecnología hoy en día.

El ruido es la piedra más grande del camino, los dispositivos de ASR necesitan sensores que filtren eficientemente todas las señales no deseadas que se generan en el medio ambiente. El ruido causa que algunas veces el sistema no pueda decodificar la señal. Para que esta tecnología sea empleada en este campo se necesita garantizar su efectividad y un buen costo.

Esta práctica pretende aplicar todo lo aprendido en el semestre para diseñar y desarrollar una aproximación de un sistema de reconocimiento de discurso automático, pues como se desarrolló en la introducción, no se puede desarrollar un ASR ideal pues es un sistema muy complejo que requiere herramientas aún desconocidas por el equipo.

Desarrollo

Utilizando la aplicación de Android *Record ASR* y un micrófono, se tomaron grabaciones de 26 individuos articulando los fonemas del idioma español. Esta aplicación nos permitió elegir una frecuencia de muestreo de 48,000 Hz y determinar que el número de canales fuera uno. Además, contaba con una función para cancelar el ruido que fue de gran utilidad. Los fonemas a articular por cada individuo se presentan en la siguiente tabla.

Tabla 1. Fonemas del idioma español con ejemplos de articulación

FONEMA	***LETRA(s)***	***EJEMPLO(s)***
a	a	c**a**s**a**, **a**ndar, **a**lma
b	b,v,w	**b**eso, **v**ida
θ (THETA)	c (ante e, i) z (ante a, o, u)	**c**ielo, **z**apato
ch	ch	**ch**imenea
d	d	**d**e**d**o, **d**ía
e	e	**e**lefante
f	f	**f**echa, **f**lecha, **f**laco
g	g (ante a, o, u ó consonante) gu (ante e, i)	**g**ato, **g**uerra, **g**uitarra, **g**ordo
(silencio)	h	**h**acha, **h**umo
i	i, y	m**i**, re**y**, mu**y**, **y**eso
j	g (ante e, i) j (a, e, i, o, u)	**j**amón, **j**efe, **j**inete, **j**oroba
k	c (ante a, o, *u* o consonante) qu (ante e, i) k	**c**aballo, **c**ocodrilo, **c**ueva, **q**ueso, **k**ilo
l	l	**l**abio, **l**ibra
ll	ll	**ll**ave, e**ll**a
m	m	**m**amá, **m**eza, milla, mosca
n	n	nopal, nieve
ñ	ñ	ni**ñ**o, a**ñ**o
o	o	oso

p	p	pelota, pipa
r	r	arena, ca**r**ibe
rr	rr, r	carro, radio
s	s	sol
t	t	tortuga, tigre
u	u	una, uva
ks, gs	x	e**x**amen, ta**x**i
y	y	yo, yema, yerno

Cada audio se clasificó de la siguiente manera:

FONEMA_EDAD_SEXO_ID.wav

Donde FONEMA es cada uno de los elementos de la tabla 1, EDAD se dividió en adulto (A), puberto (P) o niño (N), SEXO se dividió en hombre (H) y mujer (M) y ID es el número de identificación de cada uno de nuestros individuos. Esta forma de clasificar los audios nos permitirá tratar de identificar, dado un audio definido, la edad y sexo de la persona que lo está articulando.

Posteriormente, se subieron los audios a una carpeta y se trabajaron a través de Google Colab. Utilizando Python y sus librerías numpy, matplotlib.pyplot y scipy, se desarrolló una función que calcula la Transformada Rápida de Fourier de cada audio. Para ello, se aplicó una ventana de hamming, después un filtro de preénfasis y por último se calculó un LPC de orden 16 para obtener las formantes de cada fonema. Estas formantes fueron procesadas por una función que las pasaba a un Excel y las ordenaba por fonema, edad, sexo y id respectivamente.

Una vez que teníamos nuestra base de datos completa, creamos una fórmula que tomaba los valores de la base de datos, específicamente los valores de las etiquetas y tres formantes por cada fonema y calculada la media y varianza de las formantes, para crear así una base de datos estadística de los fonemas y sus formantes. Estos valores se almacenaron en tablas, para poder utilizarlas a lo largo de nuestro proyecto.

Además, se decidió crear 4 bases de datos estadísticas:

- BD_Vocales: Contiene los datos estadísticos sólo de las vocales, sin considerar edad y sexo.
- BD_Vocales_Completa: Contiene los datos estadísticos sólo de las vocales, considerando edad y sexo.
- BD Fonemas: Contiene los datos estadísticos de todos los fonemas, sin considerar edad y sexo.

- BD Fonemas_Completa: Contiene los datos estadísticos de todos los fonemas, considerando edad y sexo.

Una vez que contábamos con las bases de datos, comenzamos a trabajar con múltiples audios. Algunos de los audios con los que trabajamos articulaban las siguientes dos frases:

- “Hola Mundo”
- “Procesamiento Digital de Señales”

La frecuencia de muestreo de cada audio variaba entre dos valores, 44100 Hz y 48000 Hz. Sin embargo, todas eran a un solo canal; de no ser así, teníamos una función que nos ayudaba a hacer la conversión. Una vez que contábamos con los audios, lo primero que hicimos fue obtener su gráfica en el tiempo.

La primera aproximación con la cual tratamos de crear nuestro sistema de ASR consistió en lo siguiente. Primero, se determinó la longitud empírica de los fonemas dentro de un audio. Después, basándonos en este valor y en el estándar de tamaño de fonema que se suele usar en sistemas ASR, el cual indica que la duración de un fonema es aproximadamente de 27 ms, lo cual a una frecuencia de muestreo de 48,000 Hz representa 1296 muestras, decidimos que tomaríamos ventanas de 3500 muestras, que es un promedio entre los dos valores antes mencionados. Una vez que teníamos la señal seccionada en el tiempo, aplicamos un filtro de preénfasis, una ventana hamming y un LPC de grado 16 a cada sección. La respuesta a esta operación se comparaba con nuestra base de datos y se elegía como posible fonema aquel que cumpliera mejor los parámetros de media y varianza.

Después, se decidió utilizar un espectrograma para obtener los valores de FFT de las ventanas. Para ello, se procesó el audio aplicando un filtro de preénfasis y luego tomando su espectrograma. Este último utilizaba ventana de 1024 muestras (aproximadamente de 31 ms), tenía un overlap de 100 muestras y utilizaba un enventanamiento tipo blackman. Una vez obtenidos los resultados del espectrograma, estos eran comparados contra nuestra base de datos estadística y se elegía como posible fonema aquel que cumpliera mejor los parámetros de media y varianza.

Finalmente, se implementó el algoritmo de K-Neighbours. El desarrollo de esta técnica se realizó con dos enfoques. Ambos partían de usar los resultados obtenidos por el espectrograma, tomando cada ventana producida por el mismo y graficando este posible fonema en un plano en tercera dimensión; los valores de cada eje representan sus tres primeros formantes. Sin embargo, las metodologías difirieron en lo siguiente.

En la primera, se decidió graficar en el mismo plano todos los fonemas de nuestra base de datos sin procesar (es decir los datos crudos sin incluir medias y varianzas). Después, se calculaban los tres vecinos más cercanos al fonema que se estuviese estudiando y, con base a esto, se tomaba una decisión de cuál sería la aproximación más apropiada.
En cambio, en la segundo metodología se decidió graficar solo un valor que representase cada fonema, el cual era su centroide (para ello redefinimos las bases de datos antes mencionadas para que conservaran la media y descartaran la varianza). De esta manera, se comparaba el fonema a estudiar con los valores de la gráfica y se aproximaba a aquel

centroide al cual estuviese más cerca, teniendo una restricción de distancia máxima. Si el valor supera esta distancia máxima a todos los posibles fonemas, entonces se descarta como ruido.

Al final se optó por utilizar la última metodología planteada de k neighbors y se aplicó a nuestras 4 bases de datos mencionadas anteriormente (BD_Vocales, BD_Vocales_Completa, BD Fonemas y BD_Fonemas_Completa).

Por último, todo lo anterior fue implementado en una GUI que contaba con la siguiente funcionalidad:

- Poder seleccionar audios a analizar de los archivos de la máquina.
- Poder grabar un nuevo audio a analizar.
- Poder correr el algoritmo de ASR el cual muestre los resultados aproximados de la palabra a encontrar y grafique el audio en el tiempo, además de su espectrograma.

Resultados

Bases de datos

Una vez obtenidos los audios de todos los fonemas del español, se utilizó un código de Python para calcular las formantes de todos los audios y guardarlos en un google sheets. El código utilizado fue el mismo que ya se había empleado para calcular formantes, utilizando el filtro de preénfasis y el análisis lpc. La base de datos se encuentra en la carpeta del proyecto bajo el nombre de “BD_Formantes.csv”.

En cuanto a la base de datos original, después de hacer el mismo análisis con todos los audios con lpc de orden 16, se observaron las diferencias brutales entre algunos audios, algunas formantes eran mucho mayores que el resto o no fueron encontradas en el análisis.

Para solucionar estos problemas, se decidió eliminar las formantes dispersas para aliviar la desviación estándar. A continuación se muestran los resultados de la letra A como ejemplo.

FONEMA	EDAD	GÉNERO	ID	FORMANTE 1	FORMANTE 2	FORMANTE 3
A	N	H	13	1075.82	3218.89	9230.95
A	N	H	25	0	0	0
A	P	H	20	0	0	0
A	P	H	24	1079.4	4256.5	8649.01
A	A	H	3	834.31	3346.93	6066.97
A	A	H	4	978.87	4895.06	7619.25
A	A	H	5	808.43	2504.94	6232.16
A	A	H	7	562.4	5113.77	0
A	A	H	15	653.81	0	0
A	A	H	17	871.87	2595.0	7731.54
A	A	H	18	1016.32	3789.35	6483.82
A	A	H	23	1031.41	10210.86	0
A	A	H	26	998.56	3419.46	5262.04
A	P	M	2	1223.81	5641.17	11485.88
A	P	M	11	691.66	7163.52	11032.64
A	A	M	1	1104.16	1987.22	5043.06
A	A	M	6	813.02	7437.67	11741.09
A	A	M	10	1251.97	4501.31	6738.14
A	A	M	12	0	0	0
A	A	M	14	966.25	2805.34	7697.68
A	A	M	15	545.39	1220.18	3969.37
A	A	M	16	676.25	1572.54	4459.28

A	A	M	19	844.49	1555.93	4119.29
A	A	M	21	0	0	0
A	A	M	22	886.69	7829.21	10282.68

Al principio, no limpiamos la base de datos y eso afectó el desempeño de nuestra primera ronda de pruebas.

Primera ronda de pruebas: pruebas de media con desviación estándar

Tras investigar cómo funcionaban los ASR, previo a la práctica 9 del espectrograma, se decidió armar una base de datos estadística que guardara la media y la desviación estándar de cada formante en cada fonema.

La base de datos estadísticas resultaron un éxito en términos de la media, pues cuando checamos las formantes teóricas de muchas letras, nuestros valores coincidían. No obstante, la desviación estándar muchas veces fue excesiva.

Para las primeras pruebas, se grabó un audio diciendo “Hola Mundo” y se dividió en 27 ms con overlap de 5ms para hacer el análisis individual de cada grupo de muestras. El procedimiento consistió en calcular las formantes del grupo de muestras y comparar si la formante n estaba dentro de un rango de media más menos su desviación estándar. Si la formante se encontraba en ese rango, guardaba un diccionario con el código del fonema como un posible resultado, este análisis se hacía por cada formante, por lo que el diccionario tenía el objetivo de guardar el número de veces que coincidía el audio con la base de datos.

El diccionario tenía el siguiente formato:

dic={A_H_N:1,E_M_N:1,N_H_P:3,B_M_N:1, … ,Y_M_A:4}

Las pruebas siguieron este rumbo después de cambiar el formato al espectrograma, ya que los cambios fueron mínimos. El espectrograma también se dividió a 1024 muestras, que eran aproximadamente 27 milisegundos y se agregó un overlap de 100 muestras.

Sin embargo, estas pruebas resultaron en un fracaso, ya que como la desviación estándar era enorme en la mayoría de las formantes, el diccionario la mayor parte de los análisis regresaba números altos de coincidencias en casi todos los fonemas.

Al notar el conflicto que surgía por las desviaciones estándar, el equipo decidió limpiar la base manualmente. Al limpiar la primera formante, nos dimos cuenta lo problemático de nuestro método, ya que no teníamos datos teóricos sobre las formantes de los fonemas después de la tercera, entonces no sabíamos qué valores se podían quitar y cuáles no. Especialmente a partir de la quinta formante, la desviación estándar era enorme, y el equipo desconocía cuáles si eran válidas y cuáles no.

Tras la primera ronda de pruebas, el equipo consideró los siguientes puntos:

- La limpieza de la base de datos era imposible tomando en cuenta más de 3 formantes,
- La idea de utilizar la desviación estándar no estaba ayudando a limitar los datos, sino que complicaba el análisis,
- Era necesario hacer un análisis de distancia euclidiana para lograr mejores resultados y
- Para hacer un análisis más riguroso, podíamos bajar el número de muestras por ventana para que fueran 21 ms.

Segunda ronda de pruebas: Algoritmo K-Neighbours

Para la segunda ronda de pruebas, se implementó el algoritmo K-Neighbours, el cual calcula la distancia euclidiana hacia toda la base de datos, para así encontrar aquel con menor distancia.

Para realizar este análisis, se decidió medir distancias respecto a la base de datos original y una nueva base de datos estadística. Además, se restringió el número de formantes analizadas a sólo las primeras 3.

En cuanto a la nueva base de datos estadística, se guardaron solo las medias, para obtener los centroides en un plano de 3 dimensiones y calcular la distancia euclidiana. La base de datos tenía el formato que se presenta en la siguiente tabla.

CÓDIGO	**FORMANTE 1**	**FORMANTE 2**	**FORMANTE 3**
A_N_H	1075.82	3218.89	9230.95
A_P_H	1293.04	4424.24	8649.01

Al hacer las mediciones del K-Neighbours con cada base, obtuvimos resultados muy variados. Algunas veces, el programa regresaba letras que tenían sentido al medirlas con respecto a la base de datos completa, pero la mayoría de los análisis regresaban cosas sin sentido, el principal conflicto fue la dispersión, ya que al realizar las medidas las más cercanas regularmente eran las medidas dispersas antes que los resultados esperados.

En cuanto a la base de datos estadística se tenía el mismo problema, la dispersión de algunos datos hacía que los resultados arrojados por el programa no tuvieran mucho sentido gran parte de las pruebas, pues los valores dispersos modificaban en ocasiones gravemente al cálculo de medias.

En comparación con la primera ronda de pruebas, los resultados fueron mucho mejores, ya que el nuevo método arrojaba en algunas muestras los resultados esperados, mientras que la primera ronda de pruebas nunca regresaba resultados correctos.

Tras la segunda ronda de pruebas, el equipo consideró los siguientes puntos:

- K-Neighbours fue un algoritmo que deberíamos usar corrigiendo los parámetros de error obtenidos en las pruebas,
- Era necesaria una limpieza de la base de datos para eliminar la enorme dispersión obtenida en las mediciones y
- Los conflictos más grandes se encuentran en las consonantes, por ello debemos dividir el análisis en una impresión sólo de vocales y una impresión de todos los fonemas.

Tercera ronda de pruebas: Aproximación final

Se comenzó limpiando los datos dispersos de la base, si las formantes tenían una dispersión muy clara, se eliminaban y se ponía un cero en su lugar. Esta limpieza fue arbitraria y no aseguramos su efectividad, pero dado que tomar audios de nuevo no era opción, limpiar la base de datos manualmente fue la única solución óptima.

Al terminar la limpieza de la base de datos, el análisis de K-Neighbours con la base de datos completa quedó descartada y se decidió únicamente analizar con respecto a la nueva base de datos estadística, que tiene el mismo formato que la base de datos estadística utilizada en la segunda ronda de pruebas, el único cambio fue los datos tomados para crear esa base de datos estadística.

Para mostrar la funcionalidad del programa, el equipo decidió implementar un análisis de K-Neighbours con 4 bases de datos, las 4 tenían una base en la base de datos estadística generada: una base de datos de las vocales sin género ni edad, una base de datos de las vocales con género y edad, una base de datos con todos los fonemas sin género ni edad y la base de datos estadística que ya se mencionó.

Este análisis con respecto a 4 bases distintas permitió observar cómo funcionaba nuestro ASR y cuáles eran sus debilidades. Además, restringimos los resultados a una distancia máxima, si la distancia era mayor a 1 simplemente no se consideraban. Los resultados arrojados por cada análisis tienen el formato que se muestra a continuación.

```
Ventana: 31
Ventana: 53
CH 0.4030430487328213
F 0.7659421547824085
Ventana: 71
Ventana: 74
Ventana: 82
Ventana: 83
Ventana: 84
Ventana: 85
Ventana: 86
N 0.42595685122056426
```

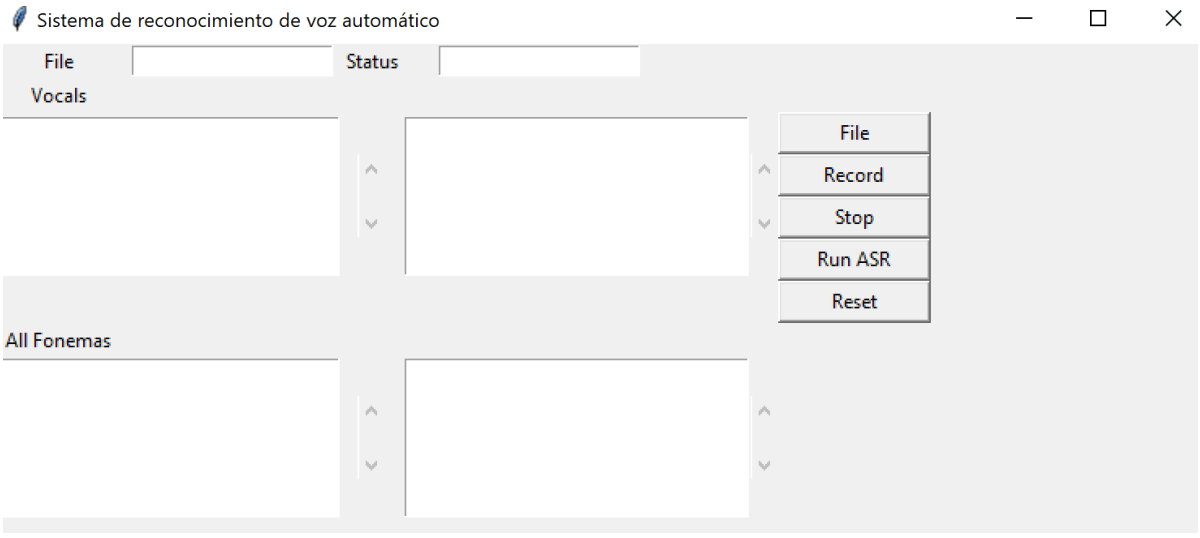
```
E 0.43170381361991134
P 0.6154881242299128
THETA 0.7267553337931034
KS 0.7717012395696722
O 0.7890415070727178
Ventana: 87
Ventana: 88
Ventana: 89
Ventana: 90
Ventana: 91
D 0.5103541329226401
```

Se puede observar que muchas veces las ventanas que sí tenían formantes, no encontraban ningún valor cercano y por eso no escribían nada. Se decidió implementar esta misma solución, con variantes mínimas en la impresión, en la interfaz gráfica. A continuación se explica los resultados de la GUI.

GUI

A continuación, se muestra una imagen de nuestra interfaz gráfica.

Imagen 1. ASR GUI



En cuanto a la creación de esta GUI, el equipo se enfrentó a varios retos. Lo primero consistía en que no conocíamos ninguna librería para el desarrollo de GUIs en Python. Lo primero que nos dimos cuenta es que las GUIs no pueden crearse en entornos virtuales y nosotros estábamos utilizando Google Colab. Ante ello, lo primero que tuvimos que hacer fue mudar nuestro código a otra plataforma, en este caso PyCharm, lo cual ocupó bastante tiempo.

Una vez hecho lo anterior, estudiamos la librería Tkinter que permite la creación de interfaces gráficas en python. Realmente la creación de la parte visual de la GUI (cómo *labels*, *buttons*, *listbox*) fue bastante fácil e intuitiva. Sin embargo, la parte que se complicó más fue cargar archivos, grabar audios y graficar dentro de la GUI.

La parte de cargar archivos y grabar audio fue difícil debido a que Tkinter no tiene como tal métodos integrados para llevar a cabo esta funciones. Debido a esto, se tuvo que crear las funciones de cero, lo cual nos tomó mucho tiempo debido a que requirió que hiciéramos una amplia investigación acerca del tema y del procesamiento de audio utilizando Python. Sin embargo, al final logramos nuestro objetivo. En cuanto a la parte de graficar archivos, la mayor complicación fue que Tkinter ofrece componentes en su GUI de gráficos para la creación de gráficos sencillos; sin embargo, la función que utilizamos para la creación del espectrograma (specgram) integra en sí una gráfica al ejecutarse la cual Tkinter no reconocía. No obstante, con un poco de investigación y algunas funciones que tuvimos que crear logramos hacer la impresión.

Podemos concluir diciendo que el mayor reto en el desarrollo de la GUI fue el poco tiempo que tuvimos para desarrollarla y lo poco que sabíamos acerca de la creación de interfaces gráficas en Python.

Simulación de nuestro sistema ASR

Para mostrar los resultados de una simulación, se optó por utilizar el audio en el que se articula lo siguiente “Procesamiento Digital de Señales”.

Imagen 2. Cargar archivo

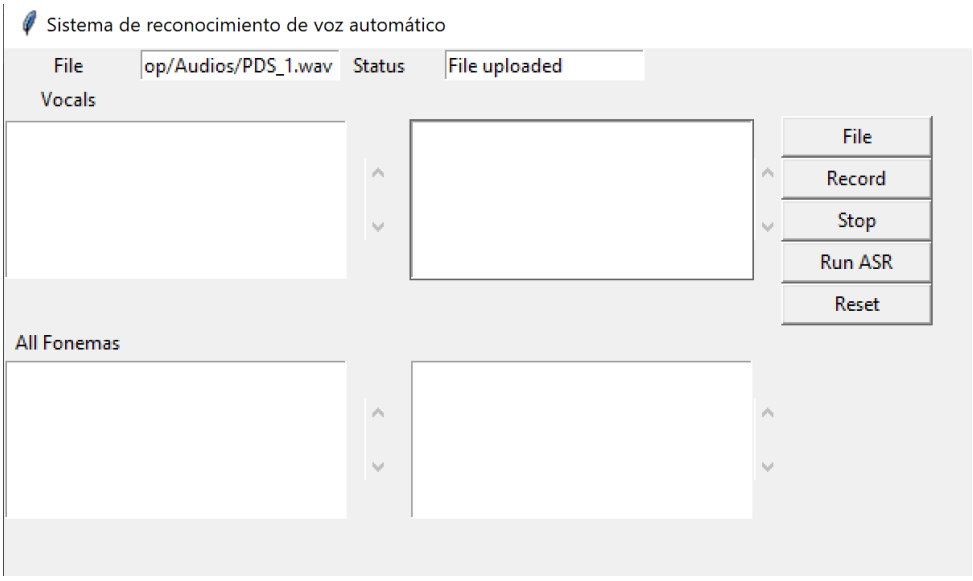
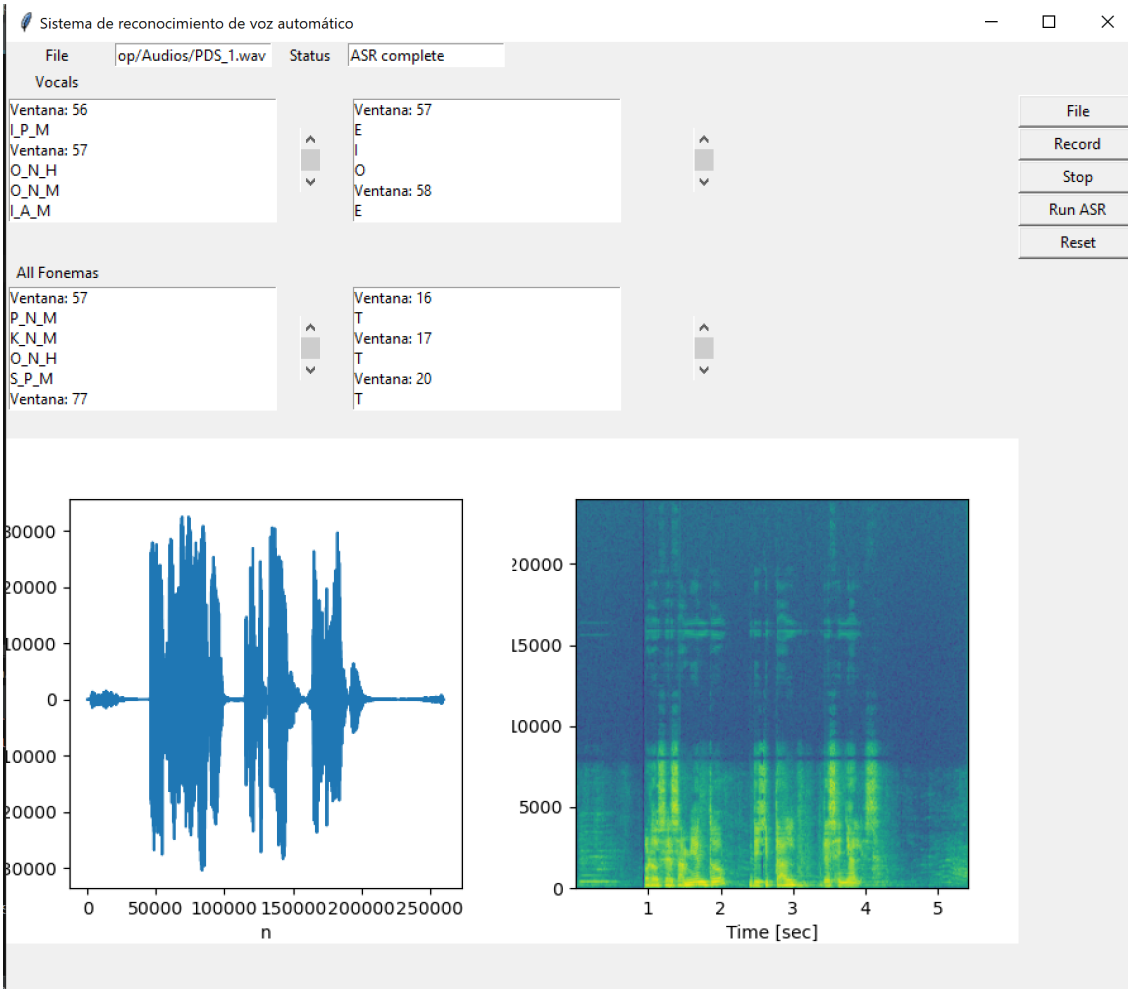


Imagen 3. Resultados ASR



En cuanto a los resultados de cada sección, estos se incluyen a continuación.

Vocales considerando sexo y género (voc1):

Ventana: 56
I_P_M

Ventana: 57
O_N_H
O_N_M
I_A_M

Ventana: 58
O_N_M

Ventana: 179
A_A_H

Vocales sin considerar sexo y género (voc2):

Ventana: 57
E
I
O
Ventana: 58

E

Ventana: 177

O

Ventana: 276

E

Fonemas considerando sexo y género (fon1):

Ventana: 57

P_N_M

K_N_M

O_N_H

Ventana: 77

KS_N_M

Ventana: 102

KS_A_H

Ventana: 151

J_N_M

Ventana: 153

I_N_M

Ventana: 179

A_A_H

Ventana: 224

THETA_A_H

Ventana: 229

KS_A_H

Ventana: 243

KS_N_M

Ventana: 246

J_N_M

Ventana: 276

P_A_H

Ventana: 277

A_P_M

T_P_M

Fonemas sin considerar sexo y género (fon2):

Ventana: 16

T

Ventana: 17

T

Ventana: 20

T

KS

Ventana: 32

T

KS

Ventana: 57

Ñ

KS

O

Ventana: 58

KS

Ventana: 63

T

Ventana:96

T

KS

Ventana:101

T

Ventana:134

T

KS

Ventana:151

T

Ventana:153

T

Ventana: 177

P

Ventana: 212

KS

T

Ventana: 222

T

K

Ventana:246

T

Ventana: 251

T

KS

Ventana: 279

P

La idea que el equipo tenía era interpretar los resultados de las cuatro bases de datos y al ver las semejanzas y las diferencias entre ellos, poder llegar a una propuesta de palabra final. Es decir, si en las tres bases de datos, alrededor de la ventana 53 noto que hay una letra en común, entonces decido que esa letra seguramente si es un fonema de la frase. No obstante, después de varios intentos notamos que nos era muy difícil implementar un código que pudiese hacer eso que queríamos. Esto se debe a que los comportamientos varían muchísimo y más entre frases distintas, entonces era muy difícil implementar que se tomaría en cuenta y que no, además de definir el número de ventanas donde podría existir un mismo fonema. Para desarrollarlo podríamos haber creado una red neuronal o algoritmos más inteligente de los que creamos en este proyecto.

Sin embargo, a continuación se muestra de manera empírica el algoritmo que queríamos desarrollar y el posible resultado que hubiésemos elegido.

	P	R	O	C	E	S	A	M	I	E	N	T	O
voc 1			V: 58 O										
voc 2													V: 177 O
fon 1	V: 57 P			V: 77 KS		V: 102 KS	V: 149 A						
fon 2				V: 58 KS								V: 153 T	
RES	P	-	O	S	-	S	A	-	-	-	-	T	O

	D	I	G	I	T	A	L
voc1						V: 179 A	
voc2							
fon1				V: 153 I		V: 179 A	
fon2					V: 177 T		
RES	-	-	-	I	T	A	-

	D	E
voc1		
voc2		
fon1		
fon2		
RES	-	-

	S	E	Ñ	A	L	E	S
voc1		V: 276 E					
voc2				V 277: A			
fon1	V: 229 S						
fon2	V:212 S						V:251 S
RES	S	E	-	A	-	-	S

De esta forma, la posible palabra que hubiésemos encontrado sería la siguiente:

P O S S A T O I T A S E A S

El siguiente paso sería tener un algoritmo que pudiese tratar de completar las palabras o encontrar un sentido a ellas, lo cual implica una dificultad mayor a lo visto en clase y tampoco pudimos desarrollarlo para el proyecto.

P r OS e SA mien TO dig ITA l de S E ñ A l e S

Además, como se puede ver tenemos ciertas bases de datos que toman en cuenta el sexo y la edad de la persona que gesticula la frase, por lo que podríamos tratar de identificarlos. La idea del algoritmo que queríamos crear era que, una vez que obtenemos nuestra palabra final, obtuvieramos también los índices de los fonemas en cuanto al sexo y a la edad. Teniendo esto, pensábamos escoger aquellos valores que más se repetían como posible edad y sexo de la persona gesticulando el audio.

El funcionamiento correcto de estas propuestas que teníamos evidentemente requerían tener una mayor y mejor base de datos.

Conclusiones

A lo largo de este semestre nuestras habilidades del procesamiento y filtrado de señales incrementaron bastante, la mejor forma de representar nuestro avance es mediante este complejo proyecto.

Al iniciar este proyecto nos faltó visualizar la tremenda complejidad que conlleva, un buen dispositivo ASR no sólo utiliza una base de datos para comparar los audios, sino que tiene una estructura mucho más compleja que necesita herramientas que aún desconocemos, como son redes neuronales o las cadenas escondidas de Markoff.

Para resolver este proyecto se intentó atacar por diferentes puntos, pero ninguno fue suficiente para llegar a un resultado esperado. Realizar un ASR es un proyecto muy ambicioso que, con nuestro nivel de conocimientos actual, es imposible resolver.

Otro punto que fue importante analizar fue la complejidad que implica crear y limpiar una base de datos. Creemos que esta es una herramienta muy importante, pero al mismo tiempo muy difícil para cualquier persona que estudie o trabaje con cualquier conjunto de datos. Creemos que esto nos llamó mucho la atención e hizo que tratemos de investigar un poco más acerca de este tema el cual tiene un campo muy amplio de aplicaciones y herramientas.

No obstante, aprendimos mucho más de lo que esperábamos, entendimos la estructura de un sistema de reconocimiento de voz y también entendimos mucho mejor el funcionamiento de la voz en cada persona, los problemas que pueden surgir ante la falta de un micrófono de alta calidad.

Al final, se le dedicó mucho más tiempo del esperado a este proyecto, alrededor de un mes, pero nos permitió aprender muchísimas cosas que no sabíamos antes y nos dio la oportunidad de ser autodidactas con muchos temas avanzados de PDS que al final fueron útiles para el proyecto.

Otro punto importante a mencionar fue como cambió la manera de trabajar del equipo por las restricciones sanitarias. La falta de contacto entre el equipo y otros equipos del salón, afectó negativamente, pues fue muy difícil dar retroalimentaciones constantes. Esto hizo que la colaboración en este proyecto quedara un poco varada.

El proyecto fue un éxito y un fracaso a la vez, fue un éxito porque se utilizaron las herramientas que se analizaron todo el semestre de forma óptima, fue un fracaso porque los resultados estuvieron muy lejos de ser lo que esperábamos al principio del proyecto.

Sin embargo, el proyecto es muy interesante, ya que como se comentó en la introducción, tiene una gran cantidad de aplicaciones, es por eso que el equipo estaría gustoso de seguir investigando para mejorar los resultados obtenidos.

Referencias

1. A. Benade. (1976). Fundamentals of Musical Acoustics, Oxford University Press. Extraído de http://hyperphysics.phy-astr.gsu.edu/hbasees/Music/vowel.html
2. A. García. (2016). Transformada Rápida de Fourier: MATLAB. [Internet]. Disponible en http://www.sc.ehu.es/sbweb/fisica3/datos/fourier/fourier_1.html
3. Multiteide. (2016). “Los espectrogramas. Parte 1: Empecemos con lo básico”. [Internet]. Disponible en http://www.multiteide.es/2016/10/656/
4. Li, J., 2020. *Automatic Speech Recognition - An Overview | Sciencedirect Topics*. [online] Sciencedirect.com. Available at: <https://www.sciencedirect.com/topics/engineering/automatic-speech-recognition> [Accessed 19 May 2020].
5. Definitions, V. and Hope, C., 2020. *What Is Voice Recognition?*. [online] Computerhope.com. Available at: <https://www.computerhope.com/jargon/v/voicreco.htm> [Accessed 19 May 2020].
6. Carreras, R., 2020. *Datos Y Estadísticas Sobre El Crecimiento Del Mercado De Asistentes De Voz*. [online] Medium. Available at: <https://medium.com/@robertocarreras/datos-y-estad%C3%ADsticas-sobre-el-crecimiento-del-mercado-de-asistentes-de-voz-d9c763af8ad1> [Accessed 19 May 2020].
7. Vajpai, J. and Bora, A., 2020. *Industrial Applications Of Automatic Speech Recognition Systems*. [online] Ijera.com. Available at: <https://www.ijera.com/papers/Vol6_issue3/Part%20-%201/O6301088095.pdf> [Accessed 19 May 2020].
8. Amin, S., 2020. *Speech Recognition Is Hard—Part 1*. [online] Medium. Available at: <https://towardsdatascience.com/speech-recognition-is-hard-part-1-258e813b6eb7> [Accessed 19 May 2020].
9. Oropeza, J., 2020. *Algoritmos Y Métodos Para El Reconocimiento De Voz En Español Mediante Sílabas*. [online] Scielo.org.mx. Available at: <http://www.scielo.org.mx/pdf/cys/v9n3/v9n3a7.pdf> [Accessed 19 May 2020].