CAPÍTULO II: Metodología de trabajo y herramientas

2.1 Desarrollo

A lo largo de éste capítulo se expondrá todo el trabajo realizado para poder desarrollar un algoritmo capaz de realizar una clasificación de características para determinar en qué grupo recae la mayor cantidad de características para así formular un sistema de reconocimiento de voz. Se empieza mostrando la selección de material y herramientas para posteriormente dar una explicación más detallada del funcionamiento del algoritmo.

2.2 Selección del Material

Al desarrollar este trabajo me aseguré de seleccionar los dispositivos, plataformas y herramientas más adecuadas para poder llevar a cabo el propósito de mi tesis de la manera más viable y confiable. Tuve que investigar y realizar pruebas para poder determinar cuáles eran los métodos, herramientas (software), y dispositivos (hardware) que cumplían las características para poder trabajar bajo las condiciones requeridas para realizar el análisis de datos deseados.

El proceso de selección de dichos componentes llevó bastante tiempo pues en muchas ocasiones tuve que seleccionar y descartar al momento de desarrollar el trabajo. Muchos de los componentes se descartaron debido a su falta de viabilidad en cuanto a costos o por compatibilidad para poder trabajar en la plataforma en que se desarrollaría el programa. Algunos ejemplos que no concordaron con el desarrollo del sistema fueron tales como Windows a 64-bits por su falta de compatibilidad con las funciones que queríamos utilizar, en muchas ocasiones las librerías se desconfiguraban y tenían que ser instaladas nuevamente, el IDE Python porque no incluía las librerías precargadas que necesitamos además de algunos problemas para ligar ciertas librerías al programa.

Otro ejemplo de dispositivo que se descartó fue el micrófono de mi teléfono iPhone de Apple y en sí el software con el que grabé las primeras pruebas. El tipo de extensión que me grababa audio el teléfono era un .m4a el cual contenía muy baja calidad de audio e imposible para las herramientas que ocupé para convertirlos en un arreglo numérico. Finalmente otro problema se presentó al momento de comprimir el archivo con el teléfono y descomprimirlo en la computadora pues éste presentaba pérdida de información. La solución a éste problema fue el ocupar un programa llamado Audacity versión 2.0.3, el cual permitía una grabación de calidad directamente al ordenador.

Al realizar la selección de material pude dividir el equipo necesario en dos bloques, estos son los siguientes:

- Herramientas de Hardware

- Herramientas de Software

2.2.1 Selección de plataforma

En realidad la selección de la plataforma o sistema operativo donde se realizará el algoritmo mucho tuvo que ver con todas las herramientas y el ambiente de programación. En mi trabajo decidí desarrollarlo en una plataforma ya conocida como Microsoft Windows 7 Profesional a 32-bits (Figura 12). Una de las razones principales para elegir trabajar a 32 bits es el hecho de que mucha de la paquetería de librerías que planeaba utilizar como herramientas estaban ya probadas para plataformas a 32 bits y existían algunas versiones “demo” o “a prueba” a 64 bits, las cuales prometían bastante, sin embargo, las reseñas señalaban incompatibilidades o problemas.

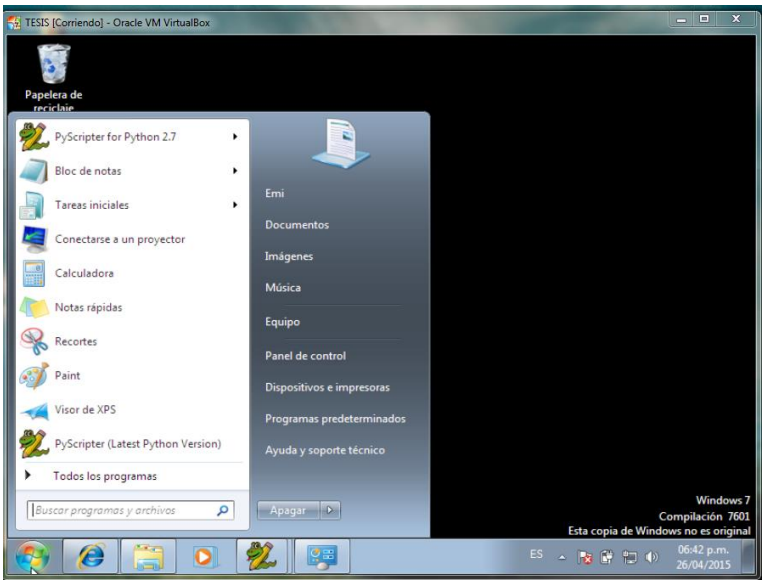

Figura 12: Ambiente de plataforma Windows 7 a 32-bits en VM de Oracle

Por esa razón, y por el hecho de no contar con una computadora con un sistema operativo a 32-bits instalé una computadora virtual. Para este caso decidí inclinarme por la "virtual machine" VirtualBox de Oracle en su versión 4.3.22 (Figura 13). Decidí trabajar con Oracle puesto que ya había trabajado en anteriores ocasiones con este software y puedo decir que tuve buenas experiencias con su desempeño.

Figura 13: VirtualBox de Oracle en su versión 4.3.22

2.2.2 Selección de dispositivo de captura de Audio

Con respecto a la selección del dispositivo de captura de audio me di a la tarea de indagar sobre micrófonos que fueran capaces de captar con una alta fidelidad en condiciones de un entorno de bajo ruido, pero con la capacidad de leer señales de ruido en cualquier dirección. Mi decisión fue entonces la de un micrófono omnidireccional capacitivo. Específicamente el modelo AT8010 de Audio Technica, el cual me manejaba ciertos valores de operación ideales para el uso que requería sin comprometer la calidad de la grabación que necesitaba. En la Tabla (Figura 14) se muestran algunos parámetros y características de éste micrófono.

Elemento	Condensador polarizado permanente de placa trasera con carga fija
Patrón Polar	Omnidireccional
Respuesta de Frecuencia	20-20.000 Hz
Atenuación Gradual de Graves	80 Hz, 12 dB/octava
Sensibilidad de Circuito Abierto	*Phantom*: –44 dB (6,3 mV) re 1V a 1 Pa Batería: –45 dB (5,6 mV) re 1V a 1 Pa
Impedancia	*Phantom*: 250 ohms Batería: 300 ohms
Nivel de Sonido de Entrada Máximo	*Phantom*: 137 dB SPL, 1 kHz a 1% T.H.D. Batería: 123 dB SPL, 1 kHz a 1% T.H.D.
Rango Dinámico (Típico)	*Phantom*: 113 dB, 1 kHz al max SPL Batería: 99 dB, 1 kHz al max SPL
Relación Señal/Ruido	70 dB, 1 kHz a 1 Pa
Requisitos de la Potencia Phantom	11-52V DC, 2 mA típico
Tipo de Batería	1,5V AA/UM3
Corriente / Vida de la Batería	0,4 mA / 1200 horas típico (alcalina)
Interruptor	Plana, atenuación (*roll-off*)
PESO	165 g (5,8 oz)

Figura 14: Características del microfóno AT8010 de Audio Technica

2.2.3 Selección del Software

El software que seleccionamos fue PyScript en su versión 2.7 (Figura 15) para el lenguaje de programación Python debido a que Python es un lenguaje de programación que más me convenció debido a sus múltiples ventajas para trabajar. A continuación, algunas de las ventajas para trabajar con Python:

- Es simple y rápido: No requiere gran estructura, unas cuantas líneas de programación y ya está.
- Ordenado y Limpio: Python mantiene un orden que es fácilmente digerible y legible para el programador o cualquier otro.
- Portable: Nos ofrece una flexibilidad para correrlo en otras plataformas, ya sea MAC OS, LINUX, o Windows.
- Librerías: Python nos ofrece una gama amplia de librerías bastante útiles de las cuales podemos aprovechar y aportar.
- Comunidad: La extensa comunidad de Python es bastante útil para poder conseguir documentación y salir de dudas en casos donde se requiera.

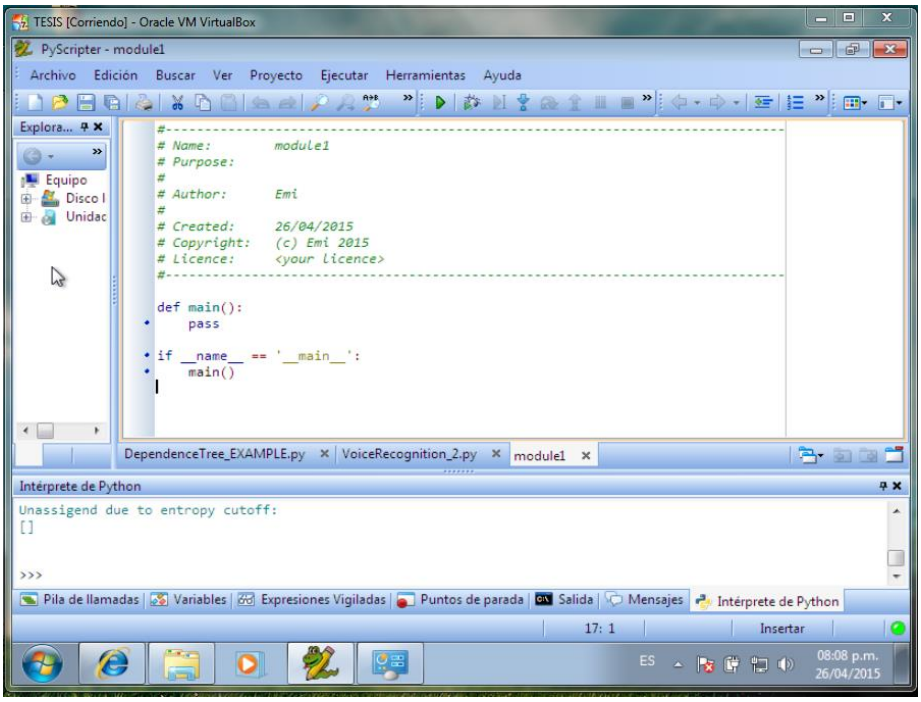

Figura 15: Python Version 2.7

Además, este lenguaje ya cuenta con las librerías que nuestro trabajo requiere para ser descargadas y utilizadas. Entre las librerías que nuestro proyecto requiere están PyWavelets para la transformada Wavelet, NumPy, Pymix y SciPy entre otras.

2.3 Herramientas

La selección de herramientas que realicé se facilitó por el hecho de que la distribución de Python con la que trabajé, ANACONDA y PyScripter, traían incorporada algunas de ellas como lo fue el caso de las librerías Numpy, SciPy, y matplotlib. Éstas herramientas las ocupo para desarrollar algunas funciones que ya vienen probadas y documentadas en las páginas oficiales de estas librerías por lo que me daba un respaldo para comprobar y confiar en los resultados esperados.

2.3.1 Numpy / Scipy

Numpy es la librería fundamental para desarrollar sistemas computacionales científicos con Python. Contiene entre muchas otras funciones:

- Un poderoso objeto de arreglos n-dimensional.
- Funciones sofisticadas (Broadcasting).
- Herramientas para integrar código de C, C++ y Fortran.
- Habilidad para trabajar con utilidades de álgebra lineal, transformada de Fourier y números aleatorios (random).

Aparte del obvio uso científico, Numpy puede también ser usado como un eficiente contenedor de datos genéricos multi-dimensionales. Pueden ser definidos tipos de datos arbitrariamente. Esto permite que Numpy se integre rápida e ininterrumpidamente con una amplia variedad de bases de datos.

Dentro de mi algoritmo Numpy juega un papel importante dentro del almacenamiento de datos que corresponden a valores numéricos importantes como en el caso de un arreglo de flotantes, donde se almacenan los coeficientes que se desfragmentan de las señales estudiadas. De esta manera Numpy nos permite guardar estos datos en un arreglo sin perderlos posteriormente. Una aplicación bastante útil ya que ocupo éstos datos en reiteradas ocasiones.

En cuanto a la librería SciPy, es una recopilación de algoritmos numéricos y “toolboxes” de dominio específico, incluyendo, procesamiento de señal, optimización, estadística y mucho más. SciPy se pronuncia “Sigh Pie” y es un ecosistema basado en Python de software de licencia libre (open source) para matemáticas, siencia e ingeniería. En particular, éstos son algunos de los paquetes básicos:

- Matplotlib: una librería bien simentada y populas que nos provee de graficaciones de calidad en 2D así como de graficación rudimentaria en 3D.
- Pandas: nos provee de alto performance y estructuras de datos de fácil uso.
- SymPy: utilizado principalmente para matemáticas simbólicas y álgebra computacional.
- iPython: una interface efectiva que te permite rápidamente procesar datos y probar ideas. El notebook de iPython trabaja en conjunto un buscador WEB, permitiéndote documentar tus resultados en una forma fácil y reproducible.
- Nose: un marco para probar el código Python.

2.3.2 Pymix

El paquete de mezclas de Python (PyMix) es una librería de disponibilidad gratuita que implementa algoritmos y estructuras de datos para una variedad amplia de aplicaciones de procesamiento de datos con modelado de mezclas básicas y extendidas. Sus características son:

- Modelado de mezclas finitas de características discretas y continuas.
- Disponibilidad amplia de distribuciones (Normal, Exponencial, Discreta, Dirichlet, Normal-Gamma, Uniforme, HMM)
- Modelado de mezclas bayesianas
- Estimación paramétrica de ML y MAP
- Estructuras independientes de aprendizaje de contexto específico.
- Parámetros de aprendizaje parcialmente supervisados.
- Estimación de parámetros para clasificación de muestras restringidas.

Dentro del desarrollo de mi programa, la librería Pymix es escencial para poder utilizar sus herramientas y utilizar funciones como lo son convertir un arreglo numpy a un DataSet (figura 16) necesario para poder meter los datos en un modelo de mezcla.

```
66
67 dataM = mixture.DataSet()
68 dataM.fromArray(dataCACD)      #57343 samples #(cA,cD)
69 m.EM(dataM,40,0.1)
70
```

Figura 16: Ejemplo de conversión de arreglo a un DataSet

Otra función que me permite crear Numpy es una mezcla de Gaussianas a partir de un Árbol de dependencias (Figura 17) el cuál se explicará más ampliamente en los temas siguientes. Sin embargo, Pymix permite recrear un modelo bayesiano que asume independencia entre las características. Esto a su vez, puede ser problemático para datos donde haya presencia de dependencias. Por otra parte la distribución Gaussiana multi-variable incluye una estructura de covarianza completa. De cualquier forma, intentar aprender de la matriz de covarianza con no más que unas pocas dimensiones, resultaría en un proceso computacional pesado (consumo de tiempo) y principalmente nos conduce a un alto riesgo de "overfitting". Definiremos el "overfitting" como una muy baja tolerancia de compatibilidad entre la muestra y los datos recuperados. Esto quiere decir que a menos que sea un muestreo perfecto, la clasificación siempre caerá dentro de la clase (muestra), excluyendo lecturas de datos muy cercanas al modelo muestra.

Entonces para poder trabajar con una estructura de árbol de dependencia será necesario trabajar con la función "ConditionalGaussianDisctribution" (Figura 17) la cual nos permitirá asignar un valor jerárquico para que un dato depende de otro, también llamado el *parent /*

children. Mitchell (2005) explica que cada nodo o característica de la red del árbol denota una variable aleatoria, los cuales son condicionalmente independientes de sus nodos no-descendientes dados sus padres inmediatos. Una distribución de probabilidad condicional está definida por P(N|Padres(N)). Una estructura ejemplo para esta función es como la siguiente.

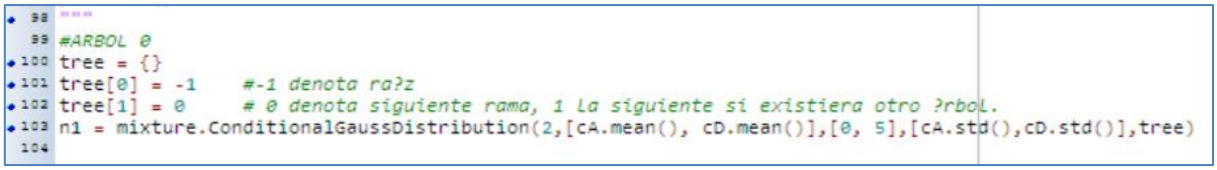

Figura 17: Ejemplo de un árbol de dependencia donde se muestra como los cD se definen con una independencia condicional de los cA. Es decir que cD son los datos *parent* y los cA los *children.*

2.3.3 Pywavelets (pywt)

PyWavelets en un software de licencia libre para la transformada Wavelet que cuenta con una interfaz sencilla, esta librería se integra con el lenguaje Python. Entre las características principales de dicha librería se encuentran:

- Transformada Directa e Inversa de Wavelet (DWT y IDWT)
- Transformada estacionaria de Wavelet
- Librería de descomposición y reconstrucción de Wavelet
- Aproximación de Wavelet y funciones de escalamiento.
- Más de setenta filtros de Wavelets integrados y soporte para Wavelets diseñadas.
- Cálculos de precisión simple y doble.
- Resultados compatibles con la Wavelet ToolBox de Matlab

Ésta librearía de Wavelets es la principal herramienta que ocupo para poder recuperar las características propias de la voz de cada persona. A partir de esta librería puedo recuperar los

coeficientes de aproximación (CA) y los coeficientes de detalle (CD) de un archivo de audio .wav el cual transformo a arreglo numérico y a partir de eso la función pywt.dwt() (Figura 18) la utilizo para poder extraer estás características y posteriormente trabajar con ellos.

```
40 fs, dataarray = wavfile.read('AnitaEmi.wav')
41 pywt.dwt(dataarray, 'db10', 'sym')
42 (cA, cD) = pywt.dwt(dataarray, 'haar', mode ='sym')
43
44 fs, dataarray2 = wavfile.read('AnitaNat.wav')
45 pywt.dwt(dataarray2, 'db10', 'sym')
46 (cA2, cD2) = pywt.dwt(dataarray2, 'haar', mode='sym')
```

Figura 18: Extracción de Coeficientes Wavelet a partir de un arreglo de flotantes

Como se menciona anteriormente, esta librería nos permite trabajar con filtros preestablecidos y en este caso yo trabajo con el filtro Haar, Daubechies y el Symlet. Sin embargo, la literatura y la documentación que se revisó, muestran que la forma Daubechies la ocupan en el análisis de señales de audio. Por esa razón decidí hacer todas las pruebas basándome en ésta forma de Wavelet.