

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

TÍTULO:

“ESTUDIO DE LOS MODELOS OCULTOS DE MARKOV
Y DESARROLLO DE UN PROTOTIPO PARA EL
RECONOCIMIENTO AUTOMÁTICO DEL HABLA”

Tesis previa a la obtención del
Título de Ingeniero de Sistemas

AUTORES:

Diana Xismenia Macas Macas
Willian Alfonso Padilla Pineda

DIRECTOR:

Ing. Vladimir Robles

Cuenca, Octubre del 2012

DECLARATORIA DE RESPONSABILIDAD:

Nosotros, Diana Xismeria Macas Macas portadora de la cédula de ciudadanía 0703848846 y Willian Alfonso Padilla Pineda portador de la cédula de ciudadanía 0104299466, estudiantes de Ingeniería de Sistemas, certificamos que los conceptos desarrollados, análisis realizados, así como los criterios vertidos en la totalidad del presente trabajo, son de exclusiva responsabilidad de los autores. Se permite a la Universidad Politécnica Salesiana la difusión del texto con fines académicos e investigativos.

Cuenca, Octubre del 2012



Willian Padilla



Diana Macas

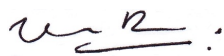
CERTIFICACIÓN:

Ing. Vladimir Robles.

Certifica:

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, realizada por la Srta. Diana Xismeria Macas Macas, y el Sr. Willian Alfonso Padilla Pineda , y por cumplir con los requisitos autorizo su presentación.

Cuenca, Octubre del 2012



Ing. Vladimir Robles B.
DIRECTOR DE TESIS

DEDICATORIA

Dedico esta tesis a mi familia que siempre me han brindado su apoyo incondicional, especialmente a mis padres: Alfonso Padilla y Digna Pineda, quienes han formado y forman una parte muy importante en cada paso que doy y han estado en todo momento junto a mí brindándome lo necesario para culminar mi carrera.

Willian Alfonso Padilla Pineda

DEDICATORIA

Esta tesis se la dedico a mi Dios quién supo guiarme por el buen camino, darme fuerzas para seguir adelante y no desmayar en los problemas que se presentaban, enseñándome a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento.

A mis padres, aunque mi padre ya no está presente en este mundo ellos siempre creyeron en mí y porque me sacaron adelante, dándome ejemplos dignos de superación y entrega, porque en gran parte gracias a ustedes, hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles.

A ti querida madre Marcela: como no agradecerte porque todo este triunfo es tuyo, por haberme apoyado en todo momento, por tus consejos, valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor. Porque el orgullo que sientes por mí, fue lo que me hizo ir hasta el final.

A mi hija Stephany por estar siempre presente, acompañándome para poderme realizar quien ha sido y es mi motivación, inspiración y felicidad.

Hermanos como no darles gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida.

A todos, espero no defraudarlos y contar siempre con su valioso apoyo, sincero e incondicional.

Diana Xismenia Macas Macas

AGRADECIMIENTOS

Agradecer en primer lugar a Dios quien siempre estuvo a mi lado, quien me guió en cada paso dado, quien me dio la fortaleza para levantarme en ciertos momentos difíciles y seguir por el camino correcto.

A toda mi familia que sin lugar a duda estuvieron pendientes y apoyándome en mi carrera, gracias por todos esos consejos que han llegado a formar parte de mis decisiones y de mi vida.

A mis amigos y compañeros que hemos pasado momentos de tristeza y alegría a lo largo de nuestra carrera universitaria que siempre permanecerán en nuestros recuerdos al igual que experiencias que nos han servido para formarnos como hombres de bien en lo académico y en lo personal.

A mi amiga y compañera de tesis, Diana con quien he trabajado durante el proceso de desarrollo de nuestro trabajo de grado.

A nuestro director de tesis el Ingeniero Vladimir Robles, quien nos brindó el apoyo y la confianza necesaria para concluir de manera correcta nuestro trabajo de grado.

Willian Alfonso Padilla Pineda

AGRADECIMIENTOS

El presente trabajo de tesis primeramente me gustaría agradecerle a mi querido Dios por darme salud, voluntad y la sabiduría para llegar a culminar una de las etapas de mi vida, porque hiciste realidad este sueño anhelado que lo empecé hace unos años atrás y ahora esta hecho realidad.

De manera muy especial a la UNIVERSIDAD POLITECNICA SALESIANA por abrirme las puertas de su prestigiosa institución dándome la oportunidad de estudiar y ser un profesional.

Como no agradecerle a mí director de tesis, Ing. Vladimir Robles por todo el apoyo, esfuerzo, dedicación, paciencia que me brindo ha logrado en mí que pueda terminar mis estudios con éxito y todo esto se debe a su conocimiento y experiencia que me supo transmitir en el momento preciso he indicado.

De una manera especial a mi amigo y compañero de tesis, Willian por el esfuerzo y la entrega que le puso al desarrollo de nuestro trabajo de grado.

De la misma manera como no agradecer a mis profesores que durante toda mi carrera profesional han aportado con un granito de arena, por sus consejos, su enseñanza y más que todo por su amistad.

Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida.

Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todas sus bendiciones.

Para cada una de esas personas presentes o no: Muchas gracias y que Dios siempre este con ustedes.

Diana Xismenia Macas Macas

ÍNDICE GENERAL

I INTRODUCCIÓN	1
1 INTRODUCCIÓN	3
1.1 Antecedentes	3
1.1.1 Técnicas más utilizadas aplicadas al Reconocimiento de Habla	5
1.2 Objetivos	7
1.2.1 Objetivo General	7
1.2.2 Objetivos Específicos	7
1.3 Alcance	8
II MODELOS OCULTOS DE MARKOV	11
2 MODELOS OCULTOS DE MARKOV	13
2.1 Introducción	13
2.2 Definición y estructura de un MOM	14
2.2.1 Definición MOM	14
2.2.2 Elementos de MOM	14
2.3 Procesos estocásticos	16
2.3.1 Clasificación de los procesos estocásticos	17
2.3.2 Matrices estocásticas	19
2.4 Arquitectura	20
2.5 Tipos de MOM	20
2.5.1 Bakis o de Izquierda a Derecha:	21
2.5.2 Ergódicos:	21
2.6 Problemas a resolver para la utilización de los MOM	21
2.6.1 Evaluación de los Modelos Ocultos de Markov	22
2.6.2 Decodificación de la secuencia de estados óptima	23
2.6.3 Entrenamiento de los Modelos Ocultos de Markov [28].	24
2.7 Aplicaciones de MOM	26
2.8 Modelo oculto de Markov para la detección automática del habla	29
2.8.1 Cadenas de Markov	29
III SELECCIÓN DE HERRAMIENTAS	41
3 SELECCIÓN DE HERRAMIENTAS	43
3.1 Introducción	43
3.2 Herramientas MOM	43
3.2.1 Hidden Markov Model (HMM) Toolbox para Matlab	43
3.2.2 HTK (Hidden Markov Model Toolkit)	44
3.2.3 CSLU Toolkit	45
3.2.4 Otras herramientas	47
3.3 Comparativa y selección	48
IV DISEÑO DE UN PROTOTIPO	53
4 DISEÑO DEL PROTOTIPO	55
4.1 Análisis del problema a resolver	55
4.2 Diseño del plan de experimentación	55

4.2.1	Pasos para el desarrollo de nuestro sistema con HTK [25]	57
4.2.2	Creación y obtención del corpus de entrenamiento	57
4.2.3	Obtención de los coeficientes cepstrales en la escala de mel, Análisis Acústico	59
4.2.4	Creación de los Modelos Ocultos de Markov	60
4.2.5	Entrenamiento de los modelos	61
4.2.6	Diccionario y Gramática	62
4.3	Pruebas con la herramienta seleccionada	63
4.4	Diseño del prototipo	66
4.4.1	Creación y obtención del corpus de entrenamiento	67
4.4.2	Obtención de los Coeficientes Cepstrales en la escala de Mel	72
4.4.3	Creación de los HMM	72
4.4.4	Inicialización de Modelos	74
4.4.5	Re-estimación de los modelos	76
4.4.6	Diccionario y Gramática	77
4.4.7	Reconocimiento de voz	79
4.4.8	Julius [33]	81
4.4.9	Palabras no incluidas en el Diccionario	88
4.4.10	Acopamiento del Sistema de Reconocimiento al Prototipo Mediante Java	89
V	RESULTADOS Y DISCUSIÓN	97
5	RESULTADOS Y DISCUSIÓN	99
5.1	Ejecución del plan de experimentación	99
5.2	Análisis de resultados	101
5.3	Especificación de mejoras planteadas a la experimentación	106
	Conclusiones	109
	Recomendaciones	111
VI	ANEXOS	117
A	ANEXOS	119
B	ANEXOS	125

ÍNDICE DE FIGURAS

Figura 1	Resumen de la clasificación de los Procesos Estocásticos[39]	19
Figura 2	Arquitectura MOM[4]	20
Figura 3	MOM de izquierda a derecha [36].	21
Figura 4	MOM Ergódicos [36].	21
Figura 5	Modelo de Markov del Clima[33]	39
Figura 6	Arquitectura HTK [34]	45
Figura 7	Arquitectura CSLU [27]	46
Figura 8	Esquema del prototipo	56
Figura 9	Pasos para el proceso de reconocimiento con HTK	57
Figura 10	Herramienta HSlab, Grabado, Etiquetado	58
Figura 11	Conversión de la frecuencia en hertzios a escala de Mel [33]	60
Figura 12	Topología básica de los HMM [40]	60
Figura 13	Proceso de entrenamiento HTK	61
Figura 14	Operación de HInit	61
Figura 15	Diagrama de flujo, Re-estimación de los HMM [25]	62
Figura 16	Red gramatical utilizada	63
Figura 17	HVite en proceso	65
Figura 18	Reconocimiento realizado	65
Figura 19	Pruebas continuas de reconocimiento	65
Figura 20	Pasos para el entrenamiento con HTK y sus herramientas	66
Figura 21	Archivo de configuración HTK	68
Figura 22	Topología Usada HMM [33]	69
Figura 23	HSlab Ejemplo Etiquetado	71
Figura 24	Archivo lista para herramienta HCopy	72
Figura 25	Ejemplo de definición de un HMM	73
Figura 26	Contenido de archivo "entrenamiento.txt"	76
Figura 27	Gramática HTK utilizada	78
Figura 28	Diccionario HTK utilizado	78
Figura 29	Archivo de configuración HTK utilizado	80
Figura 30	Listado de HMM utilizando HTK	80
Figura 31	Ejecución herramienta HVite	81
Figura 32	Reconocimiento HVite	81
Figura 33	Pasos para el proceso de desarrollo en Julius	82
Figura 34	Archivo de configuración Julius	83
Figura 35	Archivo "tiedlist", listado de Modelos utilizados	83
Figura 36	Archivo .voca	84
Figura 37	Función Para realizar el reconocimiento	85
Figura 38	Archivo .grammar	85
Figura 39	Estructura de reconocimiento Julius	86
Figura 40	Proceso de la herramienta mkdfla.pl	86
Figura 41	MOM entrenado con HTK	87
Figura 42	Función Modelos Basura	89
Figura 43	Prototipo Final	93
Figura 44	Interfaz Gráfica	93

Figura 45	Proceso de experimentación del reconocimiento de palabras	99
Figura 46	Ejecución de la herramienta Julius	100
Figura 47	Reconocimiento de una palabra mediante Julius	100
Figura 48	Pruebas continuas mediante Julius	101
Figura 49	Nivel de reconocimiento de voz obtenido en las pruebas realizadas	103
Figura 50	Nivel de Reconocimiento de Palabras No entrenadas	104
Figura 51	Resultado de pruebas con palabras no entrenadas	105
Figura 52	Interfaz de HSlab	123
Figura 53	Julius Información	126

ÍNDICE DE CUADROS

Cuadro 1	Comparativa y Selección	49
Cuadro 2	Palabras que simulan las funciones del Movimiento del Puntero	93
Cuadro 3	Palabras que permiten Activar la Función de las teclas de Dirección	94
Cuadro 4	Palabras que activan la función de ciertas teclas para el control del S.O.	94
Cuadro 5	Palabra que permite terminar el proceso de Ejecución	94
Cuadro 6	Precisión de reconocimiento de las palabras	102
Cuadro 7	Nivel de Reconocimiento entre hombres y mujeres	102
Cuadro 8	Ejemplo de pruebas con palabras no entrenadas	104
Cuadro 9	Resultado de pruebas con palabras No encontradas	105

Parte I

INTRODUCCIÓN

INTRODUCCIÓN

1.1 ANTECEDENTES

El reconocimiento automático del habla surge debido a la necesidad de contar con máquinas que se pudieran comunicar con los seres humanos de manera natural, debido a esto se ha comenzado a construir sistemas para diferentes aplicaciones, empezando por el reconocimiento de conjuntos pequeños de palabras sobre líneas telefónicas, hasta llegar a tener máquinas que nos permitan poder realizar el dictado de palabras a grandes volúmenes.

Además, existen aplicaciones que se han desarrollado para ayudar a los discapacitados y controles de entrada mediante la voz en sistemas de seguridad [19][16].

Actualmente muchas empresas y universidades han comenzado a darle mucha importancia al reconocimiento automático del habla, por tal motivo ahora se puede observar que existen aplicaciones reales que antiguamente solo se podía ver en películas de ciencia ficción [37].

Por tal razón, surge interés de parte de centros de investigación y diferentes empresas que les gusta estar siempre a la vanguardia tecnológica y tener como objetivo la creación de sistemas que incorporan la digitalización de la voz hasta poder llegar a tener una aproximación muy significativa en el reconocimiento automático del habla [37].

A continuación se va a explicar sobre los antecedentes históricos del reconocimiento automático del habla [35][19][16]:

- **Reconocimiento de una sola palabra en la década de 1920.**

En esta década se construyó un juguete llamado Radio Rex, este era un perro que se movía cuando pronunciaban su nombre, estaba configurado para que reconozca solo la vocal "e".

Aunque a veces este se movía con cualquier palabra que pronunciaban.

- **Reconocimiento de 10 dígitos aislados en la década de 1950.**

Durante la década de 1930 y 1940 hubo varias investigaciones sobre el análisis del habla. En 1952 K. Davis, R. Biddulph y S. Balashek presentan por primera vez un sistema de reconocimiento de palabras, desarrollando para los laboratorios Bell.

Dicho sistema era capaz de identificar 10 dígitos para un solo hablante mediante el reconocimiento de frecuencias.

- **Reconocimiento de palabras aisladas con varios locutores en la década de 1960.**

En esta década se comienza a emplear las propiedades acústico-fonéticas del habla. Muchos de los reconocedores electrónicos que existían para ese tiempo comenzaron a reconocer pequeños

vocabularios de palabras aisladas. Varios institutos y laboratorios japoneses comenzaron a realizar investigaciones sobre el reconocimiento automático del habla.

- **Se presentan las primeras soluciones de Reconocimiento de habla continua, vocabularios medianos y varios locutores en la década de 1970.**

El mayor logro que se dio en esta década fue proponer los primeros sistemas para el reconocimiento de habla continua. IBM inicia con proyectos de reconocimiento de grandes vocabularios.

Para las décadas de los 60 y 70 se empieza a incorporar:

- Algoritmos de extracción de características a partir de la señal acústica.
- Modelado de rasgos espectrales, incentivados por el surgimiento de la transformada rápida de Fourier.
- La aplicación de procesamiento cepstral.
- Técnicas de warping expansión o compresión temporal de señales, permitiéndonos obtener las diferentes velocidades de habla o longitud de segmentos.

- **Reconocimiento de habla continua, vocabularios grandes desde 1000 a 10000 palabras y varios locutores en la década de 1980.**

En esta etapa se empieza a realizar investigación sobre el reconocimiento del habla continua con diferentes locutores.

El principal promotor para realizar esta investigación fue la Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de Estados Unidos (DARPA).

Para mediados de los 80 se lanza la tarea de Resource Management, el cual consistía de oraciones construidas a partir de un léxico de 1000 palabras.

También se propuso la tarea Wall Street Journal, la cual contaba con un vocabulario más amplio que llegaba hasta 60000 palabras, estas eran extraídas de ese periódico.

Las dos tareas se daban mediante la transcripción del habla leída.

En esta etapa se procede a cambiar del enfoque basado en reconocimiento de patrones a métodos de modelos probabilísticos, como lo son los Modelos Ocultos de Markov.

Además comienzan a aparecer las Redes Neuronales para el reconocimiento de la voz.

- **Se dan las primeras aplicaciones en la década de 1990**

En esta década se pasa a la integración entre reconocimiento de voz y procesamiento del lenguaje natural.

Se cuenta con sistemas de dictado, y ordenadores y procesadores baratos y rápidos.

- **En la actualidad, Integración en el Sistema Operativo.**

Existen muchas herramientas, por ejemplo tenemos las siguientes:

- Sistema V2C interacción voz-radio.
- Voice Web Browsers, integración de aplicaciones por teléfono y sitios de Internet dedicados a la gestión de reconocimiento de voz.
- Aparece el estándar VoiceXML

1.1.1 Técnicas más utilizadas aplicadas al Reconocimiento de Habla

Para el reconocimiento del habla existen diversas técnicas de clasificación de patrones de voz, las cuales describiremos a continuación [14][31][18]:

- **Comparación de Plantillas o Patrones utilizando técnicas de Programación Dinámica (DTW):**

Esta técnica se enfoca específicamente en la comparación entre los patrones o plantillas que tiene el sistema con la señal acústica recibida como entrada.

Se procede a realizar la parametrización de la señal recibida y la señal de entrada se la transforma en coeficientes espectrales para ser analizada correctamente.

Después de obtener los espectros de la señal se comienza a realizar el proceso de reconocimiento mediante la comparación de patrones almacenados. Se utiliza esta técnica para resolver problemas de reconocimiento de habla continua como aislada.

El principal problema que se presenta aquí es que al momento que nosotros pronunciamos una palabra nunca va a tener una duración determinada, por lo que la plantilla nunca se ajustará con el ritmo de pronunciación, ya que depende del interlocutor.

- **Modelos Ocultos de Markov (HMM):**

Los Modelos Ocultos de Markov son considerados como autómatas finitos, los cuales están formados por varios estados que son conectados por las transiciones.

El problema que presenta es la alineación de plantillas, se procede a solucionar con el modelado estocástico, ya que esto nos ayuda a obtener una mejor señal del habla permitiéndonos tener un buen reconocimiento de habla aislada como continua.

- **Redes Neuronales:**

Las redes neuronales son sistemas compuestos por estructuras de procesamiento que operan en paralelo.

Estas obtienen el conocimiento por medio de un proceso de aprendizaje y para proceder a realizar el almacenamiento de la información se utiliza los pesos sinápticos o conexiones interneuronales.

Estas redes neuronales poseen grandes ventajas entre la cuales tenemos la capacidad de aprendizaje, tolerancia ante fallos y

capacidad de producir respuestas en tiempo real, por eso son consideradas como una de las mejores técnicas para realizar el reconocimiento automático del habla.

Estas redes presentan algunos inconvenientes o problemas que los detallamos a continuación:

- Se necesita mucho tiempo para realizar los entrenamientos.
- No tener el conocimiento de los nodos que son necesarios para abordar un problema.
- Las redes neuronales solo tienen capacidad de procesamiento espacial, lo cual causaría un inconveniente en el reconocimiento del habla ya que necesita métodos con capacidad de proceso en dos dimensiones, espacio y tiempo.

Debido a esto, se procede a combinar técnicas de Programación Dinámica así como Modelos Ocultos de Markov con estas redes, consiguiendo modelar la variable tiempo, permitiendo no sólo clasificaciones acertadas de las entradas de la red, sino también segmentación de la señal de entrada.

Con esta base se persigue analizar en profundidad los Modelos Ocultos de Markov que actualmente son un modelo dominante en el reconocimiento automático del habla.

1.2 OBJETIVOS

1.2.1 *Objetivo General*

Estudiar los modelos de Markov a fin de diseñar e implementar un prototipo de reconocimiento automático del habla.

1.2.2 *Objetivos Específicos*

- Analizar los modelos ocultos de Markov, esto es, su estructura y fundamento matemático.
- Especificar las formas de uso de los Modelos Ocultos de Markov
- Analizar la técnica de detección automática del habla basada en los modelos ocultos de Markov.
- Analizar y comparar herramientas para el reconocimiento automático del habla mediante HMM y escoger la mejor para el desarrollo de nuestro prototipo.
- Desarrollar un prototipo funcional basado en los modelos ocultos de Markov para el reconocimiento automático del habla.

1.3 ALCANCE

El enfoque del presente proyecto se centra principalmente en realizar una investigación profunda sobre el estudio de los Modelos Ocultos de Markov (MOM), los mismos que aplicaremos para realizar el Reconocimiento Automático del Habla, mediante el análisis de los procesos probabilísticos. Además, nos enfocaremos en el estudio de los 2 tipos de modelos ocultos de Markov como son los de Bakis o de Izquierda-Derecha y Ergódicos, a través de los cuales crearemos el modelo para poder analizarlo y entrenarlo.

También se procederá a efectuar el análisis de las diferentes herramientas que existen en la actualidad y de esta manera poder escoger la que nos ayude a lograr nuestros objetivos propuestos.

Posterior al análisis MOM, se realizará un estudio profundo de la herramienta escogida, enfocándola principalmente para el desarrollo de nuestro prototipo funcional basado en Modelos Ocultos de Markov que nos permitirá hacer el Reconocimiento automático del Habla.

En base al modelo construido vamos a plantear las diferentes funciones que el prototipo debe realizar y y así mismo, realizaremos las respectivas pruebas para determinar si está realizando el reconocimiento automático del habla en tiempo real de una manera correcta.

Parte II

MODELOS OCULTOS DE MARKOV

2.1 INTRODUCCIÓN

En la actualidad existen diferentes puntos de vista que se han abordado para resolver el problema del reconocimiento o clasificación de patrones en el campo del reconocimiento de la voz, pero para el desarrollo de la presente capítulo nos vamos a referir a los Modelos Ocultos de Markov (MOM¹), ya que estos tiene un enfoque estadístico. Existen varios métodos estadísticos, pero ninguno de estos expresa claramente las características estadísticas de la señal de la voz.

Por lo tanto aquí nos vamos a enfocar estrictamente en los estadísticos basados en los procesos estocásticos. Estos procesos de Markov nos proporcionan la estructura necesaria para poder realizar la modelización estadística de los procesos, para poder concretar el proceso de toma de decisiones y de esta manera que la perdida sea lo más pequeña posible.

La teoría de los MOM y su aplicación en el campo del reconocimiento de la voz no es nueva. Esta teoría aparece a finales de la década de los 60 y principios de la década de los 70 y es propuesta por Leonar E. Baum y sus colegas, que publicaron una serie de documentos acerca de este tema. La primera aplicación de procesamiento de la voz fue implementada por Baker de la CMU², Jelinek y sus colegas de la IBM³ en los años 70 [21].

En los años 90 se presenta por primera vez una herramienta de reconocimiento de voz, esto se da debido a que para la década de los 80 recién se presentaron al público e investigadores los primeros tutoriales de MOM sobre el reconocimiento de la voz [21].

Los MOM en la práctica han sido aplicados en diversas áreas de la ciencia como el análisis de imágenes, en la psicología, en el seguimiento de partituras musicales, etc. Su uso más destacado ha sido en la bioinformática y en el análisis de electroencefalogramas y otras bioseñales [4].

¹ MOM: Modelos Ocultos de Markov

² CMU: Canegie Mellon University

³ IBM: International Business Machines Corporation

2.2 DEFINICIÓN Y ESTRUCTURA DE UN MOM

En el mundo siempre se están presentando diferentes eventos, para lo cual es necesario la creación de modelos que nos permitan modelarlos. En virtud de ello nos enfocaremos en su estudio para poder conocer su comportamiento.

Para la representación de los eventos existen dos tipos de modelos, entre los cuales tenemos los determinísticos y los estocásticos.

- **Los modelos determinísticos:** Son modelos matemáticos que son establecidos por una secuencia conocida sin tener influencia del azar a través del tiempo [1].
- **Los modelos estocásticos:** Son modelos probabilísticos, que toman a la incertidumbre para realizar los cálculos, analizando los procesos que se presentan de forma aleatoria entre estados y teniendo influencia del azar [3].

De acuerdo al análisis de estos modelos nos proponemos en este presente capítulo analizar los modelos estocásticos, los que nos van a permitir simular los procesos de la voz mediante el estudio de los Modelos Ocultos de Markov.

2.2.1 Definición MOM

El modelo oculto de Markov es un conjunto finito de estados, cada uno de los cuales está asociado con una distribución de probabilidad.

Las transiciones entre los estados son administradas por un conjunto de probabilidades llamadas probabilidades de transición. En un estado en particular un resultado u observación puede ser generada, de acuerdo con la distribución de probabilidad asociada.

Además, estos son modelos estadísticos que nos permiten la modelización de los procesos o datos secuenciales, lo cual permite tener mejores representaciones de los eventos probabilísticos, que se producen de una manera aleatoria entre los estados.

La principal característica de los MOM es su doble proceso estocástico uno oculto o no observable y otro observable, se dicen que son ocultos debido a que los estados no se los puede ver directamente, pero las variables que son influenciadas por el estado si se las puede observar [21].

2.2.2 Elementos de MOM

A continuación vamos a describir los elementos que componen a los MOM [38]:

- **N:** Representa el número de estados del modelo. Estos estados se encuentran ocultos, pero se los debe de tomar muy en cuenta ya que tienen un significado físico muy importante.

Denotación:

- **Estado:** $S = \{S_1, S_2, S_3, \dots, S_n\}$
- **Estados en el tiempo:** t como q_t .

- **M:** Representa el números de observaciones que se producen. Si estas son continuas entonces M es infinito.

Denotación:

- **Símbolo de observación:** $V = \{V_1, V_2, \dots, V_M\}$
- **Observación en tiempo:** $t, O_t \in V$.
- **A = $\{a_{ij}\}$:** Matriz de transición de probabilidades, donde a_{ij} es la probabilidad de que se de la transición desde el estado i al estado j .

Denotación:

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), \quad 1 \leq i, j \leq N, \quad 2 \leq t \leq T, \quad a_{ij} \geq 0 \quad \forall_{ij}$$

Se pueden presentar casos en el que un estado puede ser alcanzado por otro estado en un solo paso, tenemos que

$$a_{ij} > 0 \quad (2.1)$$

para todo i, j . En los MOM general, se tiene que $a_{ij} = 0$ para uno o más parejas de valores (i, j) .

$$\sum_{i=1}^N a_{ij} = 1 \quad \forall_i \quad (2.2)$$

- **B = $\{b_j(k)\}$:** La distribución de parámetros de las probabilidades de observación en el estado j .

Denotación:

$$b_j(k) = P(V_k \text{ en } t \mid q_t = S_j), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \quad (2.3)$$

tal que

$$b_j(k) \geq 0, \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \quad (2.4)$$

y

$$\sum_{k=1}^k b_j(k) = 1, \quad 1 \leq j \leq N \quad (2.5)$$

- π : Conjunto de probabilidades de estado inicial $\pi = \{\pi_i\}$, siendo p_i la probabilidad de que el estado inicial de MOM sea el S_i .

Denotación:

$$\pi_i = P(q_1 = S_i) \quad 1 \leq i \leq N \quad (2.6)$$

tal que

$$\pi_i \geq 0, \quad 1 \leq i \leq N \quad (2.7)$$

y

$$\sum_{i=1}^N \pi_i = 1 \quad (2.8)$$

Establecidos los valores apropiados para N, M, A, B y los MOM se los puede utilizar para que genere una secuencia de observaciones

$$O = O_1, O_2, \dots, O_T$$

Por lo tanto, un modelo oculto de Markov se describe como:

$$\lambda = (A, B, \pi)$$

2.3 PROCESOS ESTOCÁSTICOS

Son modelos matemáticos que permiten realizar el estudio del comportamiento de un sistema dinámico de acuerdo a los eventos que produzcan las variables aleatorias a lo largo del tiempo.

El proceso estocástico es una función aleatoria que varía en el tiempo

Características [39]:

- Describe el comportamiento de una variable aleatoria mediante una adecuada distribución de probabilidad.
- El tiempo es un fenómeno aleatorio que evoluciona según un parámetro t .
- La variable aleatoria $x(t)$ es asociada a los sistemas que presentan estados definidos y observables.
- El sistema puede cambiar de estado en cualquier momento.

- $P_x(t)$ representa un sistema que tiene una probabilidad de estado asociada.

Por lo anteriormente descrito procedemos a definir al proceso estocástico de la siguiente manera:

t : Tiempo.

$x(t)$: Variable aleatoria.

P_x : Probabilidad de estado asociado.

2.3.1 Clasificación de los procesos estocásticos

Los procesos estocásticos se pueden clasificar de la siguiente manera [15][24]:

Procesos estocásticos según la memoria de la historia de estados

Se procede a tomar en cuenta la memoria que guarda el proceso de la historia de los estados anteriores por los cuales atravesó.

Para realizar el análisis de una manera correcta se procede a definir la probabilidad condicional o de transición entre estados, lo cual va a permitir analizar la evolución del proceso.

Denotación:

$$\begin{aligned} P\{X(t + \Delta t) = x_{t+\Delta t} \mid X(t) = x_t, X(t - \Delta t_1) \\ = x_t, X(t - \Delta t_2) = x_{t-\Delta t_2}, X(t - \Delta t_3) = x_{t-\Delta t_3}, \dots\} \end{aligned} \quad (2.9)$$

Siendo:

- $x_{t-\Delta t}$: un estado particular en el instante $t + \Delta t$.
 - x_t : un estado particular en el instante t .
 - $x_{t-\Delta t}$: un estado particular en el instante $t - \Delta t_1$, etc.

De lo anteriormente descrito aparecen tres tipos de procesos :

– Procesos aleatorios puros

Los procesos aleatorios puros, son procesos “sin memoria”, ya que la probabilidad de que este sistema se encuentre en un estado cualquiera $x_{t-\Delta t}$ en el instante $t + \Delta t$, se podría calcular independientemente de cuáles hayan sido los estados anteriores por los cuales paso el modelo ($x_t, x_{t-\Delta t_1}, x_{t-\Delta t_2}$, etc).

Los procesos se los representa de la siguiente manera:

$$P\{X(t + \Delta t) = x_{t-\Delta t} \mid X(t) = x_t, X(x_{t-\Delta t_1}), X(x_{t-\Delta t_2})\}$$

$$= x_{t-\Delta t_2}, \dots\} = P\{X(t + \Delta t) = x_{t+\Delta t}\} \quad (2.10)$$

– Proceso sin memoria tipo Markov

Estos procesos estocásticos son conocidos como “memoria uno”.

Son procesos que cumplen con la siguiente propiedad:

$$\begin{aligned} P\{X(t + \Delta t) = x_{t+\Delta t} \mid X(t) = x_t, X(x_{t-\Delta t_1}), X(x_{t-\Delta t_2}) \\ = x_{t-\Delta t_2}, \dots\} = P\{X(t + \Delta t) = x_{t+\Delta t} \mid X(t) = x_t\} \end{aligned} \quad (2.11)$$

La probabilidad de que el sistema se encuentre en un estado cualquiera $x_{t+\Delta t}$ en el instante $t + \Delta t$, se puede calcular si se conoce cual ha sido el estado inmediatamente anterior x_t , sin tener conocimiento de los demás estados anteriores.

Otra forma de expresar sería de la siguiente manera, dado que el estado presente del modelo x_t , el futuro $x_{t+\Delta t}$ es independiente del pasado ($x_{t-\Delta t_1}, x_t - \Delta t_2$, etc).

– Procesos con memoria

En estos procesos con memoria se toma en cuenta todos los estados anteriores por los que el sistema pasó, para poder calcular la probabilidad de que el sistema se encuentre en un estado cualquiera $x_{t+\Delta t}$ en el instante $t + \Delta t$.

Procesos de tipo Markov según la naturaleza de las variables

En este proceso nos vamos a enfocar en las características continua o discreta del espacio de estados de la variable aleatoria $X(t)$ y de parámetro t .

Los procesos de tipo Markov de acuerdo a la naturaleza del espacio de estados son:

- **Procesos de Markov:** Llamados “procesos de tipo Markov con estados continuos”. Estos se dan cuando la variable $X(t)$ representa una magnitud continua (fuerza, energía, presión), y su espacio de estados $X(t)$ debe ser un intervalo de números reales.
- **Cadenas de Markov:** Conocidas como “procesos de tipo Markov con estados discretos”. Cuando la variable $X(t)$ representa una magnitud discreta (cantidad de clientes en un sistema de atención). En este caso, el espacio de estados $X(t)$ es una secuencia finita o numéricamente infinita de enteros.

Según la naturaleza del parámetro t , los procesos de tipo Markov pueden ser:

- **Procesos o Cadenas de Markov de parámetro continuo:** las observaciones al sistema se realizan en cualquier momento del continuo ($t \geq 0$).
- **Procesos o Cadenas de Markov de parámetro discreto:** las observaciones al sistema se realizan en determinados instantes del parámetro t (por ejemplo: cada hora, cada minuto, cada día, etc).

Cuadro de resumen de lo anteriormente explicado:

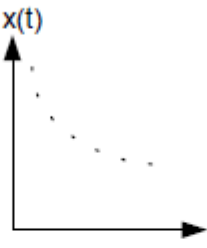
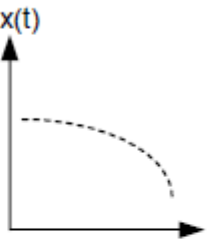
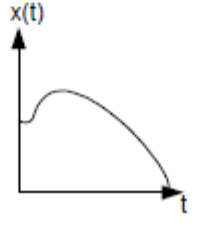
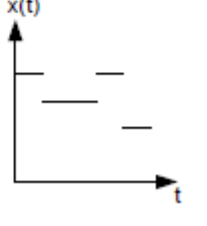
		Naturaleza del espacio de estados $X(t)$	
		Discreto	Continuo
Naturaleza del parámetro t	Discreto	Cadenas de Markov de parámetro discreto 	Procesos de Markov de parámetro discreto 
	Continuo	Cadenas de Markov de parámetro continuo 	Procesos de Markov de parámetro continuo 

Figura 1: Resumen de la clasificación de los Procesos Estocásticos[39]

2.3.2 Matrices estocásticas

Describe una cadena de Markov sobre un espacio finito de S elementos. Siendo la probabilidad de cambiar desde el estado i hasta el estado j :

$P(j | i)$, representaremos dicha probabilidad en la fila i , elemento j de la matriz: a_{ij} .

$$P = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots \\ & & \ddots & \vdots & \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots \end{pmatrix} \quad (2.12)$$

Entre las matrices estocásticas nos podemos encontrar con que estas pueden ser de los siguientes tipos [13][23]:

- **Matriz estocástica a derecha:** Se la utiliza en procesos de Markov, es cuadrada, con todos sus elementos no negativos y sus filas suman 1:
 - $\forall i, j a_{ij} \geq 0$
 - $\sum_j a_{ij} = 1$
- **Matriz estocástica a izquierda:** es cuadrada, con todos sus elementos no negativos y sus columnas suman 1:
 - $\forall i, j a_{ij} \geq 0$
 - $\sum_i a_{ij} = 1$
- **Matriz estocástica doble o bi-estocástica:** es cuadrada, con todos sus elementos no negativos y sus filas y columnas suman 1:
 - $\forall i, j a_{ij} \geq 0$
 - $\sum_j a_{ij} = 1$
 - $\sum_i a_{ij} = 1$

2.4 ARQUITECTURA

En la figura 2 presentamos de una manera general la arquitectura de un MOM [4]:

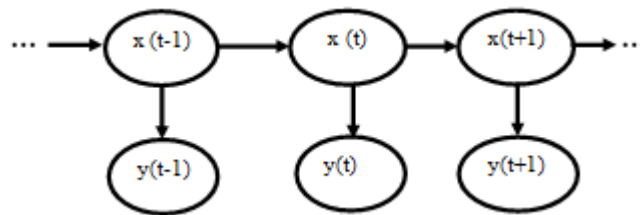


Figura 2: Arquitectura MOM[4]

- Los óvalos representan una variable aleatoria.
- La variable aleatoria $x(t)$ es el valor de la variable oculta en el instante de tiempo t .
- La variable aleatoria $y(t)$ es el valor de la variable observada en el mismo instante de tiempo t .
- Las flechas indican dependencias condicionales.

2.5 TIPOS DE MOM

Estos tipos de MOM se determinan por la cantidad de estados que va ser integrada y las transiciones que se dan entre dichos estados.

Para poder escoger el mejor tipo se debe realizar un análisis profundo de la aplicación para la que se va a utilizar los MOM.

A continuación se presentaran los tipos más comunes [36]:

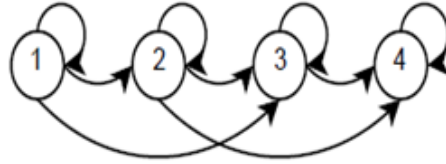


Figura 3: MOM de izquierda a derecha [36].

2.5.1 Bakis o de Izquierda a Derecha:

En este tipo de MOM solo se podrán dar las transiciones hacia delante, esto se da debido a que mientras el índice del tiempo t va avanzando, también lo hace el índice de estados. De igual forma, también se podría dar el caso que se quede en el mismo estado, este es muy utilizado para modelar las señales que varían con el tiempo como son la voz.

La probabilidad para este tipo se representa de la siguiente manera:

$$a_{ij} = 0 \quad \forall j < i$$

Esto significa que el sistema no va a tener transiciones hacia atrás o producirse una transición hacia un estado con un índice menor al actual.

2.5.2 Ergódicos:

En este tipo de MOM los estados ocultos son no nulos, no periódicos y recurrentes, pero en la aplicación práctica se los llama MOM con topología ergódica, en esta todos sus estados se conectan entre sí, o podemos decir que las posibles transiciones que se dan entre estados están habilitadas cuando $(a_{ij} > 0 \quad \forall i, j)$.

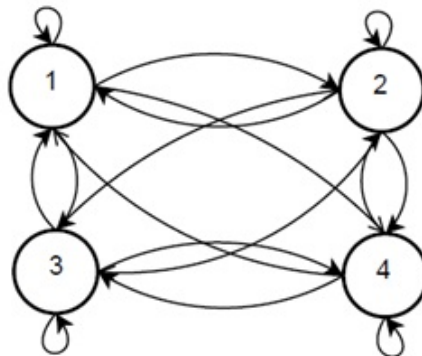


Figura 4: MOM Ergódicos [36].

2.6 PROBLEMAS A RESOLVER PARA LA UTILIZACIÓN DE LOS MOM

A continuación se da a conocer tres importantes aspectos que se debe de tomar en cuenta para aplicar los Modelos Ocultos de Markov en la vida real[30]:

2.6.1 Evaluación de los Modelos Ocultos de Markov

El proceso de evaluación se lo realiza porque se quiere verificar si la secuencia de observaciones desconocida coincide con las secuencia de observaciones modelada. *Problema:*

Se quiere determinar la probabilidad de que el modelo genere la observación acústica $P(O | \lambda)$, a partir de una observación acústica y un Modelo Oculto de Markov.

A continuación se define la probabilidad de una secuencia de observaciones dado un MOM ($P(O | \lambda)$) directamente como:

$$P(O | \lambda) = \sum_{q_1, q_2, \dots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \quad (2.13)$$

la cual es la suma de las probabilidades de la secuencia de observaciones O sobre todas las secuencias de estados posibles.

La definición de la ecuación 2.13, dentro de los términos computacionales no se lo puede aplicar en aspectos de la vida real.

De esta manera para poder realizar la determinación de $P(O | \lambda)$ se lo procede a realizar mediante los *Algoritmos Avance-Retroceso* que resultan menos costosos computacionalmente.

En la evaluación se utiliza el Algoritmo de Avance (Forward) que lo describiremos a continuación:

Algoritmo de Avance (Forward)

El Algoritmo de avance define la variable de avance $\alpha_t(i)$ como

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda) \quad (2.14)$$

donde $\alpha_t(i)$ representa la probabilidad de las observaciones O_1, O_2, \dots, O_t , y el estado S_i en el tiempo t , dado el modelo.

El cálculo de $P(O | \lambda)$ se realiza recursivamente mediante el Algoritmo 2.1:

Algoritmo de Retroceso (Backward)

Aunque en el proceso de evaluación no se utiliza este algoritmo, a continuación se va a proceder a explicar la variable de retroceso $\beta_t(i)$ la cual esta definida como:

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda) \quad (2.18)$$

es decir, obtiene la probabilidad de la secuencia parcial de observaciones desde el instante $t + 1$ hasta la observación final, dado el estado S_i en el instante t .

El definición recurrente de $B_t(i)$ se presenta en el Algoritmo 2.2:

los Algoritmos anteriormente descritos se encargan de calcular la probabilidad de una secuencia de observaciones sin importar los estados por los que se transito.

Algoritmo 2.1 Cálculo hacia adelante de la probabilidad de un secuencia de observaciones

1. Inicialización:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad i = 1, 2, \dots, N \quad (2.15)$$

2. Inducción:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad t = 1, 2, \dots, T-1, \quad j = 1, 2, \dots, N \quad (2.16)$$

3. Finalización:

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i) \quad (2.17)$$

Algoritmo 2.2 Cálculo hacia atrás de la probabilidad de una secuencia de observaciones.

1. Inicialización:

$$\beta_T(i) = 1, \quad i = 1, 2, \dots, N \quad (2.19)$$

2. Inducción:

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \dots, 1, \quad i = 1, 2, \dots, N \quad (2.20)$$

3. Finalización

$$P(O | \lambda) = \sum_{i=1}^N \beta_t(i) \pi_i b_i(o_1)$$

2.6.2 Decodificación de la secuencia de estados óptima

En este punto se utiliza el Algoritmo de Viterbi para encontrar la secuencia de estados óptima [29].

Algoritmo Viterbi

Primeramente se procede a definir la ecuación 2.21 la cual permitirá encontrar la secuencia de estados óptima $Q = q_1, q_2, \dots, q_T$, dada una secuencia de observaciones $O = O_1, O_2, \dots, O_T$ y un modelo λ :

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \dots q_t = i, O_1 O_2 \dots O_t | \lambda] \quad (2.21)$$

esta expresión $\delta_t(i)$ nos ayuda a determinar cual es la secuencia indicada para poder llegar al instante t , para esto se toma en cuenta las primeras t observaciones, las cuales terminan en el estado S_i . Teniendo la siguiente ecuación:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij} \right] \cdot b_j(O_{t+1}) \quad (2.22)$$

Despues de proceder a realizar la maximización en 2.22, toda la información se la almacena en el vector $\psi_t(j)$.

El procedimiento para decodificar la mejor secuencia de estados es el siguiente:

Algoritmo 2.3 Cálculo de la secuencia de estados más probable para una secuencia de observaciones dada.

1. Inicialización:

$$\delta_1(i) = \pi_i b_i(O_1), \quad i = 1, 2, \dots, N \quad (2.23)$$

$$\psi_1(i) = 0 \quad (2.24)$$

2. Recursión:

$$\delta_t(j) = \left[\max_{i=1,2,\dots,N} \delta_{t-1}(i) a_{ij} \right] \cdot b_j(O_t), \quad t = 2, 3, \dots, T, \quad j = 1, 2, \dots, N \quad (2.25)$$

$$\psi_t(j) = \arg \max_{i=1,2,\dots,N} [\delta_{t-1}(i) a_{ij}], \quad t = 2, 3, \dots, T, \quad j = 1, 2, \dots, N \quad (2.26)$$

3. Finalización:

$$P^* = \max_{i=1,2,\dots,N} [\delta_T(i)] \quad (2.27)$$

$$q_T^* = \arg \max_{i=1,2,\dots,N} [\delta_T(i)] \quad (2.28)$$

4. Determinación de la secuencia de estados óptima :

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1. \quad (2.29)$$

2.6.3 Entrenamiento de los Modelos Ocultos de Markov [28].

En el entrenamiento se quiere determinar $\lambda = (A, B, \pi)$, tal que $P(O | \lambda)$ sea máxima.

Partiendo de las secuencias de observaciones $O = O_1, O_2, \dots, O_T$.

Para la resolución de esto existen dos métodos que son el algoritmo de Reestimación Baum-Welch y el Algoritmo de Reestimación Viterbi.

Algoritmo de Reestimación Baum-Welch

El algoritmo Baum-Welch se define como la reestimación de los parámetros de un MOM sobre la base de otro MOM.

Primeramente se procede a definir $\xi_t(i, j)$, lo cual corresponde a la probabilidad de estar en el estado S_i en el instante t , y en el estado S_j en el instante $t + 1$, dado el modelo y la secuencia de observaciones:

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \quad (2.30)$$

Tomando en cuenta las variables forward y backward, $\xi_t(i, j)$ se puede escribir de la forma

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)} \quad (2.31)$$

Se procede a relacionar $\gamma_t(i)$ con $\xi_t(i, j)$ sumando en j , de lo cual nos da la siguiente expresión:

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (2.32)$$

Si se realiza la suma de $\gamma_t(i)$ sobre el índice del tiempo t , nos da una cantidad que vendría a ser el número esperado de veces que el estado S_i es visitado.

Esto es

$$\sum_{t=1}^{T-1} \gamma_t(i, j) = \text{número esperado de transiciones desde } S_i \quad (2.33)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{número esperado de transiciones desde el estado } S_i \text{ al estado } S_j \quad (2.34)$$

El método de reestimación de los parámetros de un MOM (π , A y B) es el siguiente:

$$\pi_i = \text{número de veces en el estado } S_i \text{ en el instante } (t = 1) \quad (2.35)$$

$$\bar{b}_j(k) = \frac{\text{número esperado de transiciones desde el estado } S_i \text{ al estado } S_j}{\text{número esperado de transiciones desde el estado } S_i} \quad (2.36)$$

$$\bar{a}_{ij} = \frac{\text{número esperado de transiciones desde el estado } S_i \text{ con el símbolo } V_k}{\text{número esperado de veces en el estado } S_j} \quad (2.37)$$

Actualmente se lo define al método de la siguiente manera $\lambda = (A, B, \pi)$ y se usa para calcular los lados derechos de las ecuaciones 2.35-2.37 permitiendo pasar a la próxima iteración.

Algoritmo de Reestimación Viterbi

Este algoritmo comienza desde una secuencia de entrenamiento $O = O_1, O_2, \dots, O_T$ y un modelo $\lambda = (A, B, \pi)$ que se inicializo aleatoriamente. Posee un sin número de restricciones las cuales estan expresadas en las siguientes condiciones 2.1, 2.2, 2.4, 2.5, 2.7 y 2.8.

En este momento se procedera a utilizar el Algoritmo Viterbi que fue explicado en la sección 2.6.2 mediante esto hara la respectiva decodificación de la secuencia de estados óptima $Q = q_1, q_2, \dots, q_T$ de O de acuerdo a λ .

Restimación de los parámetros del MOM:

$$\bar{a}_{ij} = \frac{n_{ij}}{\sum_{k=1}^N n_{ik}}, \quad i, j = 1, 2, \dots, N \quad (2.38)$$

$$\bar{b}_j(k) = \frac{u_{jk}}{\sum_{i=1}^N n_{ij}}, \quad j = 1, 2, \dots, N, \quad k = 1, 2, \dots, K \quad (2.39)$$

El vector π va a tener un valor de uno en la posición del estado inicial de la secuencia óptima Q y después se colocara ceros en las demás posiciones.

2.7 APLICACIONES DE MOM

A continuación se mencionan algunas de las aplicaciones de los Modelos Ocultos de Markov más citadas, divididas por área de aplicación [36]:

1. Biociencias

- Clasificación automática de electrocardiogramas.
- Análisis de arritmia cardíaca.
- Representación de la actividad neuronal de las cortezas visuales de simios usando diferentes estímulos visuales.
- Restauración de las grabaciones de corrientes fluyendo desde un solo canal de iones en una membrana celular.
- Modelado de series de tiempo de la cantidad de ataques epilépticos.
- Modelado de la secuencia de la cantidad de movimiento de un feto de cordero en útero a través de ultrasonido.
- Modelado estadístico, búsquedas en bases de datos y alineación de múltiples secuencias de estructuras de proteínas.
- Detección de segmentos homogéneos en secuencias de DNA.
- Aprendizaje de filogenias no singulares.
- Modelo mutagenético longitudinal de la secuencia HIV.
- Análisis a gran escala de genoma.
- Aplicaciones en la biología computacional.
- Detección de anomalías en la carga de un procesador.

- Predicción genética en DNA.

2. Epidemiología y Biométrica

- Modelado de la secuencia de comportamiento de animales bajo observación.
- Análisis de series de tiempo de homicidios relacionados con disparos de armas de fuego y suicidios en Ciudad del Cabo, África del Sur, así como de series de tiempo de nacimientos.

3. Tratamiento de imágenes

- Clasificación de texturas.
- Técnicas de representación de formas.
- Clasificación de vehículos militares en secuencias de video.
- Reconocimiento automático de palabras clave en documentos pobremente impresos.
- Clasificación de imágenes.
- Análisis de imágenes.
- Reconocimiento de objetos en tercera dimensión.
- Reconocimiento de gestos.
- Extracción de información importante de partidos de béisbol.
- Análisis de la estructura de un video de fútbol.
- Reconocimiento dinámico de expresiones faciales.
- Clasificación automática de huellas dactilares.

4. Tratamiento de señales sonoras

- Clasificación bajo el agua de señales acústicas.
- Modelo del comportamiento de objetivos y algoritmos de rastreo.
- Segmentación musical acústica.
- Clasificación de ruido ambiental.
- Reconocimiento automático de elementos de canto de aves.
- Clasificación de música folclórica.
- Detección de anomalías en la carga de un procesador.
- Extracción de sonidos bien articulados con alta confianza.
- Clasificación de patrones musicales.

5. Detección y monitoreo automático de fallas

- Detección en línea de fallas en los mecanismos de orientación de la red espacial profunda de la NASA.
- Monitoreo de la evolución del desgaste de herramientas mecánicas.
- Inspección y mantenimiento de sistemas en deterioro.
- Detección de fallas en redes de comunicación tolerantes a fallas.

6. Comunicación

- Estimación de parámetros y detección de símbolos en canales ruidosos.
- Rastreo de frecuencias de líneas y objetivos.
- Demodulación adaptativa de señales QAM en canales ruidosos.
- Ecualización ciega y semi-ciega de señales.
- Caracterización de errores de ráfaga en canales de radio de interiores.
- Filtrado Universal.
- Modelado de la propagación de canal satelital.

7. Informática

- Detección de intrusos.
- Modelado de interacción humano-computadora.
- Clasificación del tráfico de red.
- Detección de ataques de red en varias etapas.
- Tratamiento de la pérdida de paquetes para voz sobre IP.
- Codificación.
- Modelado del retraso en Internet.

8. Análisis no estacionario de series de tiempo

- Modelado de datos de actividad planetaria geomagnética.
- Detección de anomalías en la carga de un procesador.
- Modelado de series de tiempo de curvas y sismogramas.
- Análisis de la tasa real de crecimiento del producto interno bruto de Estados Unidos, en la post-guerra.

9. Control y Optimización

- Análisis de problemas de control sensibles al riesgo.

10. Climatología

- Modelar relaciones espacio-temporales entre precipitación pluvial en una serie de sitios y patrones atmosféricos sinópticos.
- Modelado de la persistencia hidroclimática.
- Segmentación de series de tiempo hidrológicas y ambientales.

11. Econometría

- Medida de la probabilidad del ciclo de punto de quiebre de un negocio.
- Aplicaciones en la venta de bienes raíces.
- Aplicaciones en las series de retorno diario.
- Modelado de series de tiempo de retorno financieras.

12. Otros

- Representación de habilidades humanas para la tele-operación del sistema robótico de una estación espacial.

- Reconocimiento visual del lenguaje de señas americano.
- Modelado de epicentros de terremotos.
- Modelado de sistemas caóticos.
- Reconocimiento de tarjetas de negocios chinas.
- Extracción de información basada en múltiples plantillas.
- Modelado del aprendizaje no paramétrico del movimiento humano.
- Reconocimiento de actividades grupales.

2.8 MODELO OCULTO DE MARKOV PARA LA DETECCIÓN AUTOMÁTICA DEL HABLA

Como describimos en la sección 2.3.1 las cadenas de Markov son conocidas como Procesos de tipo Markov en estados continuos o discretos, por tal motivo para poder realizar la detección automática del habla nos enfocaremos en el estudio de las cadenas de Markov, las cuales se aplicarán en la construcción de un modelo de reconocimiento automático del habla.

2.8.1 Cadenas de Markov

Se les da el nombre de Cadenas de Markov debido al matemático ruso Andréi Andreevitch Markov, quien las introdujo en 1907.

Cadenas de Markov de tiempo discreto

Las Cadenas de Markov son un proceso discreto en el tiempo y tienen la propiedad de que las probabilidades que describen como evolucionará el proceso en el futuro, depende del estado actual en que se encuentre el proceso, por lo tanto son independientes de los eventos ocurridos en el pasado.

En una cadena de Markov cuando la probabilidad de ir de un estado i al estado j en un paso no depende del tiempo, se las llama cadenas homogéneas y se las representa de la siguiente manera [32]:

$$P(X_n = j \mid X_{n-1} = i) \quad (2.40)$$

Probabilidades de transición

Las cadenas de Markov finitas son una secuencia de n experimentos en la que cada experimento consta de m resultados o estados posibles E_1, E_2, \dots, E_m y la probabilidad de que ocurra un resultado particular depende únicamente de la probabilidad del resultado del experimento anterior, se lo denotará de la siguiente manera [32]:

$$p_{ij} = P(X_n = j \mid X_{n-1} = i) \quad (2.41)$$

donde $i, j = 1, 2, \dots, m$. Si $p_{ij} > 0$ entonces se dice que el estado E_i puede comunicar con E_j . Los valores p_{ij} se denominan probabilidades de transición que satisfacen la condición $p_{ij} > 0$:

$$\sum_{j=1}^m p_{ij} = 1 \quad (2.42)$$

para cada $i = 1, 2, \dots, m$. Por medio de los valores obtenidos se procede a combinarlos y formar una matriz de transición T de tamaño $m \times m$:

$$T = [p_{ij}] \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix}$$

Probabilidad de transición en n pasos $p_{ij}^{(n)}$

$p_{ij}^{(n)}$ es la probabilidad condicional de que la variable aleatoria X , empezando en el estado i se encuentre en el estado j después de n pasos, y se expresa de la siguiente manera [32]:

$$p_{ij}^{(n)} = P(X_n = j \mid X_0 = i)$$

A continuación procedemos a expresar la ecuación que representa el paso de un estado i al estado j en n pasos, para $n \geq 2$, cuando la cadena haya pasado por uno de los m posibles estados en la etapa $n - 1$

$$p_{ij}^{(n)} = \sum_{k=1}^m P(X_n = j, X_{n-1} = k \mid X_0 = i)$$

Procedemos a realizar una sustitución para expresar en una igualdad los siguientes sucesos A, B, C :

Llamamos $A \rightarrow (X_n = j)$

Llamamos $B \rightarrow (X_{n-1} = k)$

Llamamos $C \rightarrow (X_0 = i)$

Sabemos que:

$$P(A \cap B) = P(A \mid B) * P(B)$$

$$P(A \cap B \mid C) = P(A \mid B \mid C) * P(B \mid C)$$

$$P((A \cap B) \mid C) = P(A \mid (B \cap C)) * P(B \mid C) \quad (2.43)$$

Procedemos a remplazar y esto nos da como resultado la ecuación de **Chapman-Kolmogorov**:

$$\begin{aligned}
 p_{ij}^{(n)} &= \sum_{k=1}^m P(x_n = j, X_{n-1} = k | X_0 = i) \\
 p_{ij}^{(n)} &= \sum_{k=1}^m P(x_n = j | X_{n-1} = k | X_0 = i) P(X_{n-1} = k | X_0 = i) \\
 p_{ij}^{(n)} &= \sum_{k=1}^m P(x_n = j | X_{n-1} = k) P(X_{n-1} = k | X_0 = i) \\
 p_{ij}^{(n)} &= \sum_{k=1}^m p_{kj}^{(1)} p_{ik}^{(n-1)} = \sum_{k=1}^m p_{ik}^{(n-1)} p_{kj}^{(1)} \quad (2.44)
 \end{aligned}$$

La matriz de transición de n pasos se puede obtener de remplazar $n = 2, 3$.

$$[p_{ij}^2] = [p_{ij}^{(1)} p_{kj}^{(1)}] = T^2 \quad (2.45)$$

quedando de la siguiente manera:

$$[p_{ij}^n] = T^n \quad (2.46)$$

Clasificación de los estados en las Cadenas de Markov

A continuación vamos a explicar los estados que se pueden presentar en las cadenas de Markov [32]:

Estado absorbente

Se dice que un estado es absorbente si es cero la probabilidad de hacer una transición fuera de este estado.

Por lo tanto, una vez que el sistema hace una transición hacia un estado absorbente, permanecerá en él siempre.

Un estado E_i es absorbente si

$$p_{ij} = 1$$

$$p_{ij} = 0 (i \neq j, j = 1, \dots, m)$$

en la i -ésima fila de T .

Estado recurrente

Es un estado tal, que una vez que el proceso ha estado en él, existe la seguridad de que volverá.

$f_j^{(n)}$ = Probabilidad de que la primera visita al estado E_j ocurra en la etapa n .

$p_{jj}^{(n)}$ = Probabilidad de que se produzca un retorno en el n -ésimo paso.

Podemos deducir que:

$$p_{jj}^{(1)} = f_j^{(1)}$$

$$p_{jj}^{(2)} = f_j^{(2)} + f_j^{(1)} p_{jj}^{(1)}$$

$$p_{jj}^{(3)} = f_j^{(3)} + p_{jj}^{(1)} p_{jj}^{(2)} + f_j^{(2)} p_{jj}^{(1)} \quad (2.47)$$

Se da la probabilidad de que un retorno en el paso tres es igual a la probabilidad de un primer retorno en el paso 3.

Así, en general:

$$p_{jj}^{(n)} = f_j^{(n)} + \sum_{r=1}^{n-1} f_j^{(r)} p_{jj}^{(n-r)} \quad (2.48)$$

Se puede expresar en términos de f_j^n

$$f_j^{(1)} = p_{jj}^{(1)}$$

$$f_j^n = p_{jj}^{(n)} - \sum_{r=1}^{n-1} f_j^{(r)} p_{jj}^{(n-r)} \quad (2.49)$$

La probabilidad de regresar en algún paso al estado E_j es:

$$f_j = \sum_{n=1}^{\infty} f_j^{(n)} \quad (2.50)$$

Si $f_j = 1$, entonces seguro que se regresa a E_j y se denomina a E_j **estado recurrente**.

Probabilidad de primera pasada en general

Se refiere al número de transacciones que hace el proceso al ir de un estado i a un estado j por primera vez. Denotamos por $f_{ij}^{(n)}$ la probabilidad de que el tiempo de espera para el primer paso de la

cadena por el estado j , saliendo del estado i , sea igual a n , con $n \geq 1$, quedando:

$$f_{ij}^{(n)} = P(X_n = j, X_{n-1} \neq j, \dots, X_1 \neq j \mid X_0 = i) \quad (2.51)$$

Entonces,

$$f_{ij}^{(1)} = P(X_1 = j \mid X_0 = i) = p_{ij}$$

$$f_{ij}^{(2)} = P(X_2 = j, X_1 \neq j \mid X_0 = i) = \sum_{k \neq j} P(X_2 = j, X_1 = k \mid X_0 = i) = \sum_{k \neq j} p_{ik} f_{kj}^{(1)}$$

\vdots

$$f_{ij}^{(n)} = \sum_{k \neq j} p_{ik} f_{kj}^{(n-1)} \quad (2.52)$$

Podemos generalizar f_j cuando se accede a j desde un estado i cualquiera:

$$f_{ij} = \sum_{n=0}^{\infty} f_{ij}^{(n)} \quad (2.53)$$

Estado transitorio

El estado es transitorio cuando un proceso entra al estado E_j nunca regresará a ese mismo estado.

$$f_j = \sum_{n=1}^{\infty} f_j^{(n)} < 1 \quad (2.54)$$

Estado ergódico Un estado ergódico es aquel que es recurrente, no nulo y aperiódico.

Estos estados son de suma importancia para poder realizar la clasificación de cadenas y para probar la existencia de distribuciones de probabilidad límite.

Clasificación de las cadenas de Markov

A continuación se va a describir las diferentes propiedades de las cadenas [32]:

Cadenas irreducibles

Las cadenas son irreducibles cuando desde cualquier estado E se puede acceder a cualquier otro, además aquí todos los estados se comunican entre si. La propiedad de estas cadenas es que todos sus estados son del mismo tipo y tienen el mismo periodo.

Conjuntos Cerrados

Para que sea un conjunto cerrado se dan condiciones que son:

- Ningún estado que este dentro de C va a ser alcanzado por otro que este fuera.
- Que el estado que este dentro de C va a ser alcanzado por cualquier otro estado que este en C .

Para que esto se de se tiene que cumplir con lo siguiente:

$$p_{ij} = 0 \quad \forall E_i \in C, \forall E_j \notin C \quad (2.55)$$

Cadenas Ergódicas

Una cadena ergódica describe de forma matemática un proceso en el cual es posible avanzar desde un estado hasta cualquier otro estado, no es necesario que esto se dé en un sólo paso, pero debe ser posible para que cualquier resultado sea logrado independientemente del estado presente.

Distribuciones estacionarias

Sea E el conjunto de estados de una cadena de Markov con matriz de transición T . Una distribución $p = [p_i]$ donde $i \in E$ se dice que es estacionaria o de equilibrio si

$$p = pT \quad (2.56)$$

Alternativamente, se puede escribir como

$$\pi_j = \sum_{i \in E} \pi_i T_{ij} \quad (2.57)$$

Teorema

Dada la matriz de transición T de una cadena de Markov finita, aperiódica e irreducible con m estados, existe, entonces una única solución al sistema de ecuaciones

$$\pi_j = \sum_{i=1}^m \pi_i T_{ij} \quad (2.58)$$

para todo $j = 1, \dots, m$, de modo que $\sum_{i=1}^m \pi_i = 1$.
Además, la solución viene dada por

$$\pi_i = \lim_{n \rightarrow \infty} T_{ij}^{(n)} \quad (2.59)$$

para todo $i, j = 1, \dots, m$.

Tiempos medios de recurrencia

Sea X_n una cadena finita, irreducible y aperiódica, entonces, $\forall j \in E$ el tiempo medio de recurrencia al estado j , u_j , se relaciona con p_j así:

$$\pi_i = \lim_{n \rightarrow \infty} T_{ij}^{(n)} = \frac{1}{u_j} \quad (2.60)$$

Observaciones

Esto indica que cuanto mayor es el tiempo medio de recurrencia del estado j , menor es la probabilidad de que nos encontremos en ese estado j , cuando pase un tiempo suficientemente grande.

Por otro lado, u_j se puede calcular como

$$u_j = \sum_{n=1}^{\infty} n f_j^{(n)} \quad (2.61)$$

En el caso de que se tenga una cadena de Markov no finita, aperiódica e irreducible se cumple el teorema anterior, aunque se requiere que todos sus estados sean recurrentes positivos.

Probabilidades de transición límite

En general, dada una matriz en forma canónica P se puede estudiar qué valores tomaría

$$P^\infty = \lim_{n \rightarrow \infty} P^n \quad (2.62)$$

donde P es la matriz de transición de una cadena de Markov finita, no necesariamente irreducible y aperiódica.

La matriz P , en su forma canónica, tiene la siguiente expresión

$$P = \begin{pmatrix} P_1 & & & \\ & \ddots & & 0 \\ & & P_r & \\ L & & & Q \end{pmatrix}$$

Casos:

1. Cada una de las submatrices $P_i, i = 1, \dots, r$ está asociada a r cadenas irreducibles, y para este tipo de cadenas se tiene una distribución límite que es estacionaria, esto es, \forall_i

$$\pi_j = \lim_{n \rightarrow \infty} P_{ij}^{(n)} \quad (2.63)$$

2. En los estados transitorios las distribuciones límite asignan probabilidad 0, luego

$$\lim_{n \rightarrow \infty} P_{ij}^{(n)} = 0 \quad (2.64)$$

\forall_j transitorio.

3. Para i transitorio y j recurrente,

$$\lim_{n \rightarrow \infty} P_{ij}^{(n)} = \lim_{n \rightarrow \infty} f_{ij} P_{jj}^{(n)} = f_{ij} \lim_{n \rightarrow \infty} P_{jj}^{(n)} = f_{ij} \pi_j \quad (2.65)$$

donde

$$f_{ij} = \sum_{k=0}^{\infty} f_{ij}^{(k)} \quad (2.66)$$

Cadenas de Markov de tiempo Continuo[32]

En la sección 2.8.1 se tenía como índice discreto del tiempo $n = 0, 1, 2, \dots$ y se presentaban muchas propiedades.

Las cadenas de Markov en tiempo continuo $t \geq 0$, aquí es difícil definir la distribución condicionada, por lo cual decimos que $X_t, t \geq 0$, es una cadena de Markov si para cualquier $0 \leq s_0 < s_1 < \dots < s_n < s$ y sus posibles estados i_0, \dots, i_n, i, j se tiene la siguiente expresión:

$$P(X_{t+s} = j) | X_s = i, X_{s_n} = i_n, \dots, X_{s_0} = i_0 = P(X_{t+1} = j | X_s = i)$$

esto quiere decir que no depende de un estado pasado para predecir el futuro solo se toma en cuenta el estado actual.

De la expresión anterior tenemos que la probabilidad de ir de i en el tiempo s hasta j en el tiempo $s + t$ depende solamente de t , osea de las diferencias de tiempos.

Ejemplo

Sea $N(t), t \geq 0$, un proceso de Poisson con tasa λ , y sea Y_n una cadena de Markov discreta con probabilidad de transición, digamos, $u(i, j)$.

Este proceso definido como $X_t = Y_{N(t)}$ es una cadena de Markov en tiempo continuo, podemos decir, X_t procede a realizar un salto de acuerdo a $u(i, j)$ en cada llegada de $N(t)$.

La falta de memoria exponencial se la procede a seguir de manera intuitiva. Tenemos que $X_{n=i}$, se la trata de una manera independiente a lo que ocurriese en el pasado independientemente, entonces el tiempo que se da hasta hasta el próximo salto se divide de manera exponencial con tasa λ , de modo que la cadena irá al estado j con probabilidad $u(i, j)$.

Probabilidad de transición para $t > 0$ es definida de la siguiente manera:

$$p_t(i, j) = P(X_t = j | X_0 = i)$$

como $N(t)$ se distribuye como una Poisson con media λt , tiene la forma

$$p_t(i, j) = \sum_{n=0}^{\infty} e^{-\lambda t} \frac{(\lambda t)^n}{n!} u^n(i, j)$$

donde $u^n(i, j)$ es la potencia n -ésima de la probabilidad de transición $u(i, j)$.

Se puede decir que la probabilidad de estar en j en el tiempo t , es igual a que aparezcan n llegadas de $N(t)$ por la probabilidad de transición en $N(t) = n$ etapas : $u^n(i, j)$.

Probabilidades y tasas de transición

Las probabilidades de transición satisfacen las ecuaciones de Chapman-Kolmogorov, que para el caso continuo son [41]:

$$p_{s+t}(i, j) = \sum_k p_s(i, k) p_t(k, j)$$

Definimos tasa de transición de i a j , y la denotamos por $q(i, j)$, al valor

$$q(i, j) = \lim_{t \rightarrow 0} \frac{p_t(i, j)}{t}$$

el cual se interpreta como la intensidad con que la cadena pasa de i a j .

Cálculo de la probabilidad de transición

Se trata de calcular la probabilidad de transición p_t a partir de las tasas de salto q .

Para ello se usa la ecuación de Chapman-Kolmogorov considerando $k = i$ y calculando el siguiente límite [32]:

$$\lim_{h \rightarrow 0} \frac{p_{t+h}(i, j) - p_t(i, j)}{h} = p'_t(i, j)$$

donde

$$p_{t+h}(i, j) = \sum_k p_h(i, k) p_t(k, j)$$

A travez de la definición de las tasas de salto

$$q(i, j) = \lim_{h \rightarrow 0} \frac{p_h(i, j)}{h}$$

para $i \neq j$, y operando, se presenta la siguiente relación:

$$p'_t(i, j) = \sum_{k \neq i} q(i, k) p_t(k, j) - \lambda_i p_t(i, j)$$

De manera simplificada se presenta la siguiente matriz:

$$Q = (i, j) = \begin{cases} q(i, j) & \text{Si } j \neq i \\ -\lambda & \text{Si } j = i \end{cases}$$

Denominada generador infinitesimal de la cadena.

La suma de las filas de Q es 0, ya que $\lambda_i = \sum_{j \neq i} q(i, j)$. Los elementos fuera de la diagonal son no negativos y los de la diagonal son no positivos.

Probabilidades límite ³²

En a cadenas de Markov discretas se estudia el comportamiento a largo plazo de la cadena aquí también se lo hace.

Se supone que la cadena es irreducible, así todos los estados se comunican, es decir para todo i, j se tiene que existen i_1, i_2, \dots, i_j tal que $q(i, i_1), q(i_1, i_2), \dots, q(i_j, j)$ son todos positivos.

Así, se obtiene que si una cadena de Markov X_t es irreducible y tiene distribución estacionaria π , entonces

$$\lim_{t \rightarrow \infty} p_t(i, j) = \pi(j)$$

Como en tiempo continuo no existe un primer $t > 0$, se necesita una condición más fuerte: se dice que π es una distribución estacionaria si $\pi p_t = \pi$ para todo $t > 0$.

Esto es muy complicado de comprobar porque implica conocer todas las p_t que no son, a menudo fáciles de calcular.

Por ello, se hace en términos de la matriz Q , esto es:

$$Q = (i, j) = \begin{cases} q(i, j) & \text{Si } j \neq i \\ -\lambda & \text{Si } j = i \end{cases}$$

donde $\lambda_i = \sum_{j \neq i} q(i, j)$ es la tasa total de transiciones a partir de i .

Se tiene que π es una distribución estacionaria si sólo si $\pi Q = 0$.

Esto se puede ver de manera intuitiva, sustituyendo en la condición anterior el valor de Q , de modo que la condición $\pi Q = 0$ queda como

$$\sum_{k \neq j} \pi(k) q(k, j) = \pi(j) \lambda_j$$

Ejemplo

MODELO DE MARKOV DEL CLIMA

Supongamos que, una vez al día, observamos el clima y le asignamos uno de entre estos 3 estados[33]:

Estado 1 – Lluvioso.

Estado 2 – Nublado.

Estado 3 – Soleado.

Usaremos un modelo de Markov simple de 3 estados para predecir el clima. Debemos conocer la probabilidad de que cambie el clima entre días. Esto lo proporciona la matriz de transición de estados:

$$A = \begin{bmatrix} 0'4 & 0'3 & 0'3 \\ 0'2 & 0'6 & 0'2 \\ 0'1 & 0'1 & 0'8 \end{bmatrix}$$



Figura 5: Modelo de Markov del Clima[33]

También podemos definir unas probabilidades iniciales, o simplemente fijar que el primer día que medimos está soleado. Con estos datos podemos calcular, por ejemplo, la probabilidad de que en los próximos 7 días, el tiempo esté “Soleado (hoy) – Soleado – Soleado – Lluvioso – Lluvioso – Soleado – Nublado – Soleado”.

Para ello, definimos la secuencia observada O como $O = 3, 3, 3, 1, 1, 3, 2, 3$. Simplemente computamos la probabilidad, usando (3.3):

$$\begin{aligned} p(O | \text{Modelo}) &= p(3, 3, 3, 1, 1, 3, 2, 3 | \text{Modelos}) = \\ &= p(3) \cdot p(3 | 3) \cdot p(3 | 3) \cdot p(1 | 3) \cdot p(1 | 1) \cdot p(3 | 1) \cdot p(2 | 3) \cdot p(3 | 2) = \\ &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} = \\ &= 1 \cdot (0'8) \cdot (0'8) \cdot (0'1) \cdot (0'4) \cdot (0'3) \cdot (0'1) \cdot (0'2) = \\ &= 1'536 \cdot 10^{-4} \end{aligned}$$

Permanencia en un estado

Gracias al modelo, también podemos responder a la pregunta: ¿cuál es la probabilidad de que el sistema permanezca en el mismo estado durante exactamente d muestras?

La secuencia observada es: $O = i_1, i_2, i_3, \dots, i_d, j_{d+1} \neq i$, por lo que calculamos:

$$p(O \mid \text{Modelo}, q_1 = i) = (a_{ii})^{d-1} \cdot (1 - a_{ii}) = p_i(d)$$

Aquí, hemos definido $p_i(d)$ como la función densidad de probabilidad de permanecer d muestras en el estado i . Gracias a (3.4), podemos calcular el número esperado de muestras que el sistema permanecerá estable en un estado concreto:

$$\bar{d}_i = \sum_{d=1}^{\infty} d \cdot p_i(d) = \sum_{d=1}^{\infty} d \cdot (a_{ii})^{d-1} \cdot (1 - a_{ii}) = \frac{1}{1 - a_{ii}}$$

el número medio de días que el clima permanecerá soleado será:

$$\bar{d}_3 = \frac{1}{1 - a_{33}} = \frac{1}{1 - 0'8} = 5$$

Parte III

SELECCIÓN DE HERRAMIENTAS

SELECCIÓN DE HERRAMIENTAS

3.1 INTRODUCCIÓN

Actualmente existen varias herramientas que fueron desarrolladas para el reconocimiento automático del habla, por lo cual en el presente capítulo vamos a realizar un análisis de estas para poder determinar cual de ellas es la mas óptima para trabajar en el desarrollo de nuestro prototipo.

Principalmente nos vamos a enfocar en la investigación y descripción de las herramientas que trabajen con Modelos Ocultos de Markov, ya que hoy en día estos modelos son muy reconocidos por su eficacia y precisión en reconocimiento de patrones de la voz.

Después de obtener las características más relevantes de cada una de las herramientas investigadas vamos a realizar un cuadro comparativo, el cual nos va a permitir escoger la herramienta que se adapte a las necesidades que nos impusimos para el desarrollo de nuestro prototipo.

3.2 HERRAMIENTAS MOM

A continuación vamos a describir varias herramientas que nos permiten realizar el reconocimiento automático del habla y escoger la que nos brinde mejor soporte para poder realizar nuestro prototipo.

3.2.1 *Hidden Markov Model (HMM) Toolbox para Matlab*

Esta herramienta es una biblioteca de funciones M-Files, esta desarrollada en Matlab y fue creada en el año de 1998 por Kevin Murphy [7].

En la actualidad solo funciona bajo el sistema Operativo Windows.

Esta herramienta posee funciones que nos ayudarán a resolver los tres problemas básicos asociados con un MOM que son:

- Problema de entrenamiento.
- Problema de evaluación.
- Descubrir el problema de estado oculto mediante el procedimientos forward-backward y el algoritmo de Baum-Welch.

Además posee otro tipo de funciones que son:

- La cuantificación vectorial (VQ)
- El algoritmo de K-medias.

Nos proporciona dos niveles de interfaz de usuario:

- La interfaz de alto nivel que nos brinda una interfaz gráfica de usuario (GUI), permite definir un MOM, resolver los problemas típicos y además no presentan los resultado de una manera gráfica y numérica.
- La interfaz de bajo nivel es el código fuente que contiene varias funciones de Matlab, esta nos permite modificar el código.

3.2.2 HTK (*Hidden Markov Model Toolkit*)

Se trata de un conjunto de herramientas portátiles que nos sirve para manipular y construir Modelos Ocultos de Markov, aunque también puede modelar cualquier tipo de serie temporal de datos. Fue creada por el departamento de Ingeniería de la Universidad de Cambridge; sin embargo, fue diseñada principalmente para construir modelos basados en el procesamiento de señales de habla [26] [25].

En la actualidad, es usado para:

- Reconocimiento y síntesis de voz
- Reconocimiento de caracteres y formas gráficas
- Análisis de vibraciones mecánicas,
- Además a sido utilizado en el análisis del ADN humano, etc.

Esta herramienta se la puede utilizar en diversas plataformas o sistemas operativos, tales como:

- Unix
- Linux
- Windows XP
- DOS.

Para realizar el procesamiento de la voz utiliza varios algoritmos como son los siguientes:

- Algoritmo de Viterbi
- Algoritmo de Baum Welch
- Algoritmo esperanza-maximización o algoritmo EM
- Algoritmo Forward-Backward.

Además se puede utilizar herramientas Gaussianas continuas o discretas, y puede implementar otra clase de parametrización de la señal como es la predicción lineal (LPC).

Consiste en un conjunto de programas, módulos y librerías escritas en lenguaje de programación C.

Entorno Operativo de HTK

El entorno de HTK consta de 34 elementos entre los cuales tenemos los siguientes: HShell, HMen, HMath, HSigP, HLabel, HLM, HNet, HDict, HVQ, HModel, HWave, HParm, HAudio, HGraf, HUtil, HTrain, HAdapt, HRec, HResult, HParm.

Cada uno de módulos tiene su función específica para trabajar con los Modelos Ocultos de Markov.

Por ejemplo, nos permiten realizar la entrada y salida de datos y la interacción con el sistema operativo mediante el módulo HSHELL.

Además, nos permiten crear diccionarios mediante HDICT.

También tiene una interfaz que soporta formatos de archivo múltiples, permitiendo importar datos desde otros sistemas, etc.

Arquitectura de software htk

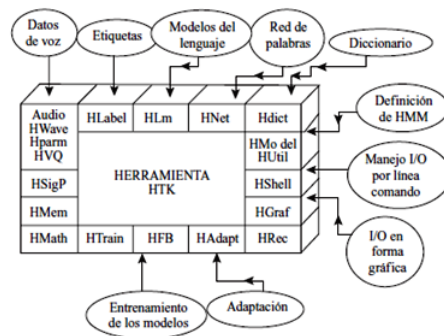


Figura 7: Arquitectura HTK [34]

3.2.3 CSLU Toolkit

Desde los años 90 se comienza con el desarrollo de esta herramienta por el Center for Spoken Language Understanding (CSLU) de Oregon Health and Science University [20].

El CSLU Toolkit fue creado para ayudarnos a realizar el desarrollo rápido y flexible de aplicaciones sobre el reconocimiento del habla y además permitirnos crear un entorno para realizar investigaciones sobre la tecnología de reconocimiento de voz.

En la actualidad, el kit de herramientas CSLU Toolkit sólo se puede instalar Windows y provee varios módulos para realizar el análisis de la voz.

Esta herramienta nos permite realizar como usuario lo siguiente:

- Realizar la manipulación de archivos WAV
- Permite realizar la extracción de las características de los fonemas
- Nos permite realizar el Entrenamiento mediante Redes Neuronales Artificiales y Modelos Ocultos de Markov (MOM)

- Creación de sistemas de reconocimiento de voz.

Además posee los siguientes componentes principales que son :

- Un conjunto de librerías que contiene módulos de tecnologías de punta
- Además cuenta con un shell de programación llamado CSLUsh el cual posee librerías escritas y desarrolladas en lenguaje de programación C y TLC
- Un ambiente de desarrollo rápido de aplicaciones (RAD).

Arquitectura de CSLU toolkit

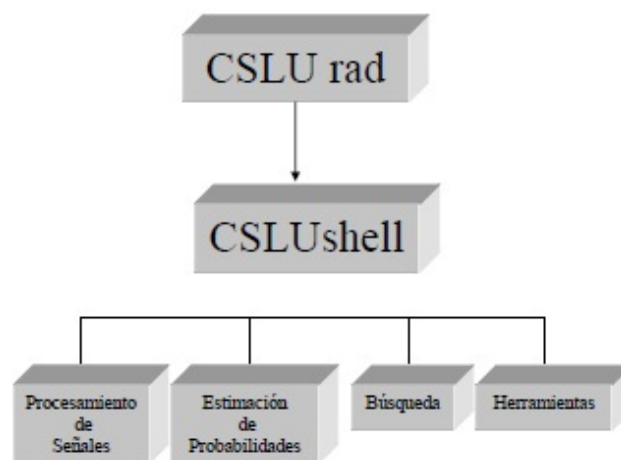


Figura 8: Arquitectura CSLU [27]

Esta herramienta nos permite realizar lo siguiente:

- Reconocimiento de voz: Se puede realizar mediante las Redes Neuronales y Modelos Ocultos de Markov .
- Síntesis de texto a voz: Para realizar esto utiliza el sistema de síntesis tts (Text to Speech) de festival.
- Animación facial: Se realiza mediante una herramienta llamada Baldi.
- Lectura de lenguaje, etc

Las técnicas de procesamiento de señales que utiliza en CSLU Toolkit para realizar la obtención de parámetros son las siguientes:

- Perceptual Linear Predicción (PPP)
- Mel Frequency Cepstral Coefficients(FCC)
- Normalización de la energía y supresión de ruido.

3.2.4 Otras herramientas

Mediante la investigación realizada se ha llegado a conocer que existen varias herramientas para realizar el reconocimiento automático del habla, aparte de las herramientas que con anterioridad describimos. A continuación procederemos a explicar algunas de ellas [26] [2] [11]:

Dragon Naturally Speaking

Este es uno de los primeros programas desarrollado y vendido por Nuance Communications, para realizar el reconocimiento del habla en ordenadores.

Las funcionalidades de este software son las siguientes:

- Se puede realizar el dictado de palabras y se la transforma en texto
- Permite tener el control del ordenador por medio de comandos de voz
- Se puede utilizar el texto para convertirlo en voz.

Este funciona bajo la plataforma de Windows y Mac OS, está desarrollada bajo el lenguaje de Visual Basic y utiliza Modelos Ocultos de Markov.

También permite a los usuarios crear documentos y mensajes de correo electrónico, completar formularios y realizar las tareas mediante la voz.

Sphinx

Este software fue desarrollado en la Universidad de Carnegie Mellon, nos ofrece un grupo de sistemas de reconocimiento de voz, permitiéndonos utilizar modelos ocultos de Markov y modelos estadísticos basados en gramáticas.

Además se encuentra disponible para todas las personas que quieran seguir desarrollando algunas aplicaciones, ya que su licencia de código es completamente libre para poder realizar cualquier modificación.

Existen cuatro versiones de Sphinx, las tres primeras están escritas en C y la última se encuentra escrita en Java.

Este framework es muy flexible, modular y permite ser incorporado a otras aplicaciones.

Los componentes de sphinx son el Sphinxtrain, para entrenamiento de los modelos, y el Sphinx Decoder para reconocimiento.

El Sphinxtrain

Este componente posee un conjunto de programas, pero su compilación solo se la puede realizar en dos sistemas: Linux y Alpha.

Nos permite trabajar con varios modelos acústicos, con topologías MOM de izquierda a derecha.

Alpha primer procesador que introduce la arquitectura de 64 bit

El Sphinx decoder

Este componente tiene algunas limitaciones, ya que no utiliza modelos de lenguaje como bigramas o trigramas, solo nos permite trabajar con modelos acústicos de 3 a 5 estados.

Los algoritmos que se pueden utilizar son los de Baum-Welch o Viterbi.

Janus Recognition Toolkit (JRTk)

Janus posee una arquitectura muy flexible, orientada a objetos que permite realizar una configuración de los componentes de una manera fácil y rápida, como es el caso de topologías MOM, secuencias de entrenamientos sin la necesidad de modificar código fuente.

Esta herramienta tiene un conjunto de utilitarios que se enfocan en el reconocimiento del habla, fue desarrollada en los laboratorios de sistemas interactivos de Carnegie Mellon University y Karlsruhe Institute of Technology.

Todo su código se encuentra bajo C y su interfaz se la maneja con Tcl/Tk, este es un entorno basado en scripts permitiendo poder construir el reconocedor de voz de acuerdo a las características que queramos que haga, además desarrollar e implementar nuevos métodos.

Janus tiene técnicas para realizar el pre-procesamiento, modelado acústico, y la búsqueda, pero todo esto lo realiza mediante Modelos Ocultos de Markov.

Tcl lenguaje dinámico de programación
Tk conjunto de herramientas de interfaz de usuario

3.3 COMPARATIVA Y SELECCIÓN

A continuación vamos a presentar una tabla comparativa de las características principales de las herramientas que analizamos con anterioridad, para poder determinar cuál es la herramienta que nos sirve para desarrollar el prototipo de reconocimiento del habla.

Software de reconocimiento de voz	Modelos	Licencia	Plataforma	Algoritmos que utiliza	Creación y manejo de redes gramaticales	Módulo de Manejo de codebook de vectores cuantificados	Arquitectura flexible y autosuficiente	Realiza etiquetado de señales
Hidden Markov Model (HMM) Toolbox para Matlab	Modelos Ocultos de Markov	Licencia MIT	Microsoft Windows	Algoritmo Forward - Backward, Algoritmo de Baum - Welch	No permite la creación y el manejo de redes gramaticales	No posee un módulo de Codebook de vectores cuantificados	No posee	No realiza
HTK (Hidden Markov Model Toolkit)	Modelos Ocultos de Markov	Código abierto	Microsoft Windows, MacOS y GNU/Linux	Algoritmo de Viterbi Belch Algoritmo esperanza - maximización o algoritmo EM Algoritmo Forward - Backward	SI permite la creación y el manejo de redes gramaticales	Si posee un módulo de Codebook de vectores cuantificados	Si posee	Si realiza
CSLU Toolkit	Modelos Ocultos de Markov	Código abierto	Microsoft Windows y GNU/Linux	Algoritmo de Viterbi Algoritmo Forward - Backward	No permite la creación y el manejo de redes gramaticales	No posee un módulo de Codebook de vectores cuantificados	No posee	No realiza
Dragon NaturallySpeaking	Modelos Ocultos de Markov	Privativo	Microsoft Windows	-----	No permite la creación y el manejo de redes gramaticales	No posee un módulo de Codebook de vectores cuantificados	No posee	No realiza
Sphinx	Modelos Ocultos de Markov	Código abierto	Microsoft Windows, MacOS y GNU/Linux	Algoritmo de Viterbi Algoritmo de Baum Welch	No permite la creación y el manejo de redes gramaticales	No posee un módulo de Codebook de vectores cuantificados	No posee	No realiza
Janus Recognition Toolkit(JRTk)	Modelos Ocultos de Markov	Código abierto	GNU/Linux	-----	No permite la creación y el manejo de redes gramaticales	No posee un módulo de Codebook de vectores cuantificados	No posee	No realiza

Cuadro 1: Comparativa y Selección

Después de haber realizado la respectiva investigación de cada una de las herramientas y verificar que beneficios nos ofrecían llegamos a la conclusión de que la que nos ofrece mejores prestaciones para poder desarrollar el prototipo es la herramienta HTK.

Debido a que nuestro principal objetivo es conseguir una herramienta que nos permita realizar el entrenamiento de la voz mediante Modelos Ocultos de Markov pero sin que dependa del locutor, escogimos dicha herramienta porque actualmente está catalogada como la mejor, ya que ha logrado tener buenos reconocimientos de la voz.

Además es importante considerar que esta desarrollada bajo código abierto, lo cual permite realizar modificaciones y de esta manera poder acoplarla a las expectativas que nosotros nos planteamos para poder realizar el reconocimiento de la voz.

Comparada con las otras herramientas, esta posee una gran cantidad de algoritmos, los cuales nos van a permitir realizar un entrenamiento con mayor precisión debido a que trabaja mediante la estadística de probabilidad, además nos permite crear modelos de acuerdo a la palabra que nosotros queramos entrenar.

En cambio las otras herramientas tenían algunos problemas debido a que algunas eran de carácter privativo y no nos facilitaban los códigos para nosotros poder estudiarlos y realizar el entrenamiento de acuerdo a nuestras expectativas.

HTK es adaptable al tipo y formato de dato que queramos modelar lo que nos da una gran flexibilidad y permite el diseño de distintos tipos de reconocedores.

También posee varios módulos de librerías, permitiéndonos una de ellas poder llegar a etiquetar cualquier base de datos lingüística.

Las herramientas Janus Recognition Toolkit(JRTk), Dragon NaturallySpeaking, Sphinx, CSLU Toolkit, Hidden Markov Model (HMM) Toolbox para Matlab, su estructura no les permite crear Redes gramaticales las cuales nos sirven de mucha ayuda para poder crear las secuencias de palabras para que el prototipo pueda reconocerlas.

Además las herramientas anteriormente descritas no tienen su propio módulo de codebook de vectores cuantificados a diferencia de Htk que si lo posee y de esta manera realizar la cuantificación vectorial de los ficheros de audio.

Parte IV

DISEÑO DE UN PROTOTIPO

DISEÑO DEL PROTOTIPO

Dentro de este capítulo vamos a detallar el proceso de desarrollo de nuestro prototipo, tomando en cuenta las diferentes fases para llegar a su conclusión, ya que se emplearon dos herramientas a más de HTK, como son Julius y clases de Java para la adaptación del sistema completo para el usuario final.

En primera instancia se detallarán todos los pasos relacionados con la configuración y diseño en HTK, es decir, procesos y utilitarios empleados.

De igual manera se explicará el uso de Julius, pues se trata de una herramienta que nos brinda mucha ayuda para el funcionamiento en tiempo real, de igual manera se indicará su modo de trabajo, parámetros, compatibilidad, etc.

Finalmente, mediante el lenguaje de programación Java, podremos adaptar el reconocimiento de voz a un prototipo con funciones específicas para el control del ordenador.

4.1 ANÁLISIS DEL PROBLEMA A RESOLVER

Lo primordial es hacer un uso correcto de las herramientas que trae HTK, para que el reconocimiento automático de ciertos comandos de voz sea óptimo.

Posteriormente, se debe adaptar los modelos entrenados a Julius, para finalmente implementar mediante Java el prototipo con el cual, gracias a los comandos entrenados, vamos a tener la posibilidad de controlar el computador mediante nuestra voz.

4.2 DISEÑO DEL PLAN DE EXPERIMENTACIÓN

Objetivo del experimento

La experimentación se llevará a cabo con el fin de crear y manipular los MOM para que mediante estos modelos sea posible crear un sistema de reconocimiento automático del habla.

Es decir, nos reconozca ciertos comandos de voz y mediante ello tener un control básico de nuestro sistema operativo.

Es importante recalcar que nos hemos planteado obtener un nivel de efectividad muy bueno en cuanto al reconocimiento de voz.

Respuesta de interés a analizar

Las respuestas experimentales que serán analizadas en este trabajo, están enfocadas a la efectividad de reconocimiento de voz de nuestro prototipo.

Para lo cual, se debe tomar en consideración los siguientes aspectos:

- **Calidad de las grabaciones de voz de nuestro corpus de entrenamiento**

Nuestro corpus de entrenamiento debe contener grabaciones de voz, con sus respectivas etiquetas, de muy buena calidad, es decir, una pronunciación correcta y con un ambiente de grabación aceptables, es decir un nivel bajo de ruido.

- **Archivos de configuración**

Para la utilización de ciertos comandos HTK, como HSLab, HCopy, y HVite es necesaria la utilización de archivos de configuración que se los envía como parámetro para cada una de ellas y según los parámetros existentes dentro de estos archivos, la herramienta va a realizar la tarea designada.

- **Género de las personas que aportaron con las grabaciones de sus voces.**

Dentro de nuestro corpus de entrenamiento, un aspecto que también se debe tomar en cuenta, es el género de cada una de las personas de las que se tomó las grabaciones, es decir, si estas personas fueron en su mayoría hombres, el reconocimiento va a ser mucho mas eficiente con voces de hombres, y viceversa, entonces para obtener un sistema que posea un nivel de reconocimiento aceptable tanto para hombres como para mujeres, el corpus de entrenamiento debe poseer muestras de voces masculinas y femeninas.

- **Creación y Entrenamiento de los modelos**

La creación y el entrenamiento de los modelos se lo debe realizar adecuadamente con un análisis previo de las necesidades y objetivos planteados para el prototipo.

- **Hardware utilizado**

Del hardware utilizado para el desarrollo y pruebas de reconocimiento va depender mucho la efectividad de nuestro prototipo, por ejemplo, el micrófono que se utilice para la grabación del corpus y de igual manera en el momento de las pruebas, debe estar en un correcto funcionamiento, al igual que una computadora en un buen estado.

Procedimientos de la experimentación

Para realizar nuestro prototipo hemos seguido ciertos pasos basándonos en cada una de las herramientas utilizadas: HTK, Julius y Java.

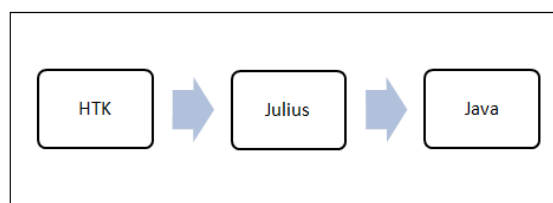


Figura 9: Esquema del prototipo

En la Figura 9, se puede observar un esquema del desarrollo del prototipo. Sin lugar a duda, el primer paso es el más importante, pues necesitamos conocer el funcionamiento de HTK en un ámbito correcto para poder trabajar con esta herramienta.

Si no existe un reconocimiento efectivo de los patrones de voz, no se podría pasar al siguiente paso, obligándonos a trabajar a fondo con HTK para nuestro sistema.

4.2.1 Pasos para el desarrollo de nuestro sistema con HTK [25]

Para la primera fase, que es el desarrollo mediante HTK, seguimos 5 pasos importantes e indispensables como se puede observar en la Figura 10.

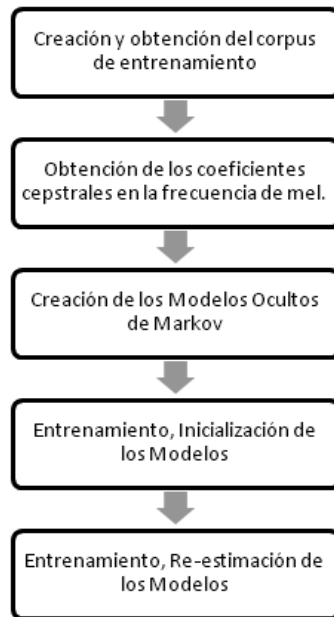


Figura 10: Pasos para el proceso de reconocimiento con HTK

Cumpliendo con estos 5 pasos obtendremos los modelos entrenados de cada una de las palabras y podremos pasar a la siguiente fase de Julius o de lo contrario si queremos realizar pruebas completas únicamente con HTK continuamos con el apartado 4.2.6.

4.2.2 Creación y obtención del corpus de entrenamiento

Como paso inicial debemos tener nuestra base de datos o corpus de entrenamiento para proceder al desarrollo del prototipo de reconocimiento. Por ello debemos crear un corpus específico para nuestro sistema de acuerdo a las necesidades planteadas.

En primer lugar, obtuvimos nuestro corpus de entrenamiento con la ayuda de 9 personas, 5 hombres y 4 mujeres y pero con la diferencia que se tuvo la facilidad de obtener por parte de los hombres mayor cantidad de muestras para cada palabra, es decir, la mayoría de muestras existentes en el corpus de entrenamiento pertenecen a voces masculinas, pero el hecho de que el corpus contenga voces de hombres y mujeres nos ayudará a que el sistema responda de mejor manera, pues mientras mayor sea la cantidad de muestras de voz empleadas para el entrenamiento mayor será la efectividad del prototipo.

La base de datos consta de 800 muestras de voz en su totalidad, tratando de poseer muestras de diferentes tonalidades para que pudiera existir un reconocimiento más globalizado.

Estas muestras han sido revisadas una a una, pues se requiere que todas tengan una calidad aceptable, ya que si son grabaciones de baja calidad el sistema presentará confusión al momento del reconocimiento.

Grabación de palabras

Para este paso inicial hemos utilizado la herramienta HSLab, puesto que en primer lugar debemos grabar muestras de voz para el entrenamiento posterior.

En nuestro caso fueron tomadas muestras de voz de las siguientes palabras: Clic, Arriba, Abajo, Derecha, Izquierda, Enter, Siguiente, Teclado, Cerrar, Gracias.

Estas fueron las palabras de las cuales se tomaron un número de grabaciones que oscila entre 20 y 50 muestras de cada una para nuestro corpus de entrenamiento, todas fueron grabadas de la misma manera, como se indica en la Figura 10, es decir:

silencio inicial + palabra + silencio final

Etiquetado de señales de grabación

Con la misma herramienta HSLab podemos realizar el etiquetado respectivo para cada una de las señales, lo que se hace mediante esto es asignar a manera de una variable cada una de las partes de la señal, es decir, describir con un nombre partes de la onda.

Las etiquetas utilizadas han sido, para silencio inicial y final la etiqueta “sil” y para la palabra, el nombre completo de cada una de ellas

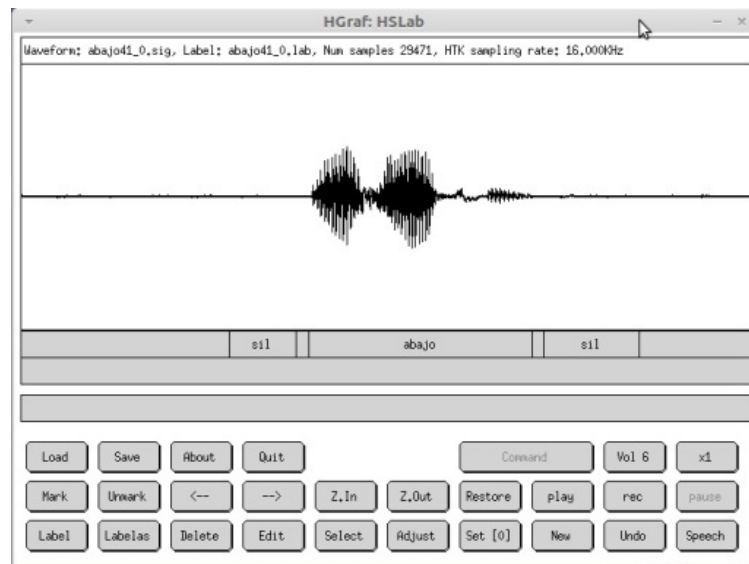


Figura 11: Herramienta HSLab, Grabado, Etiquetado

4.2.3 Obtención de los coeficientes cepstrales en la escala de mel, Análisis Acústico

Otro paso importante que se debe tratar antes de realizar cualquier proceso, es un análisis acústico, ya que existe un problema, pues las herramientas de HTK no nos permiten procesar las ondas de voz como tales.

Por ello, debemos realizar una conversión sobre nuestras muestras de voz para poder elaborar los procesos necesarios para el entrenamiento.

En este paso es muy importante un archivo de configuración que se utiliza para ciertas operaciones con las herramientas HTK.

Mediante este archivo se establecen parámetros de configuración según lo requerido, principalmente con nuestra herramienta HCopy, la cual nos sirve para realizar una conversión de archivos de audio (muestras iniciales grabadas) a vectores de coeficientes acústicos que específicamente son nuestros coeficientes cepstrales en la escala de Mel, que se trata de un tipo de archivo con extensión mfcc.

De esta forma, estamos solucionando lo antes mencionado, es decir, con nuestros nuevos archivos mfcc podemos realizar el análisis y procesos necesarios para nuestro sistema de reconocimiento de voz.

Cabe mencionar que el archivo de configuración se lo puede modificar de acuerdo a nuestras necesidades, existe una gran cantidad de parámetros o variables que establecen distintos modos de operación.

La Escala de frecuencia Mel [33]

Existen escalas de frecuencia en las que el oído humano se comporta de manera más uniforme frente a las señales de sonido que recibe, una de ellas es la escala Mel, que se relaciona con la frecuencia en hertzios mediante la siguiente expresión:

$$f_{[mel]} = 1125 \ln \left(1 + \frac{f_{[Hz]}}{700} \right) \quad (4.1)$$

Esta relación se puede observar en la Figura 12

El uso de la escala de Mel se puede decir que modela la percepción de un oído humano, mejorará la efectividad de reconocimiento.

Los coeficientes Cepstrum en la Escala de Mel (MFCC's) [25]

Estos coeficientes han dado excelentes resultados y un rendimiento superior en cuanto al reconocimiento de voz, a comparación de otros métodos de parametrización.

Para el cálculo MFCC normalmente se usa un determinado número de filtros triangulares paso-banda (Un filtro paso banda es un tipo de filtro electrónico que deja pasar un determinado rango de frecuencias de una señal y atenúa el paso del resto).

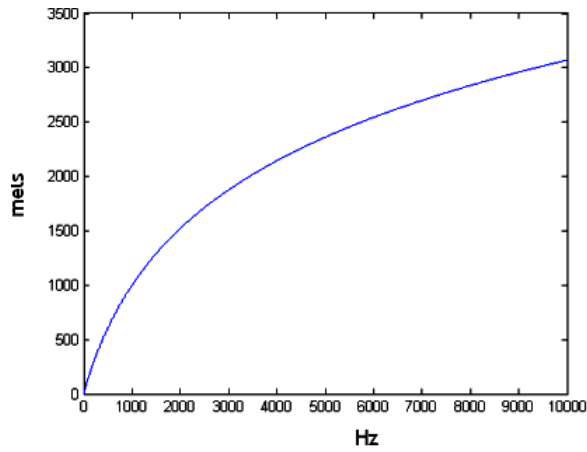


Figura 12: Conversión de la frecuencia en hertzios a escala de Mel [33]

Estos filtros están equiespaciados en la escala de Mel de frecuencias. La salida de cada filtro se puede decir que representa la energía de la señal dentro de la banda de paso de dicho filtro.

4.2.4 Creación de los Modelos Ocultos de Markov

Un paso de suma importancia es la creación de los modelos para nuestras palabras, se debe establecer un modelo oculto de Markov para cada palabra, esto se lo puede realizar con cualquier editor de texto en un archivo común.

En primera instancia se debe establecer un modelo a seguir para nuestros procesos, es decir, establecer una topología con la cual vamos a trabajar para la definición de nuestros modelos.

La topología se refiere a [25]:

- Número de estados
- Matriz de transición de estados

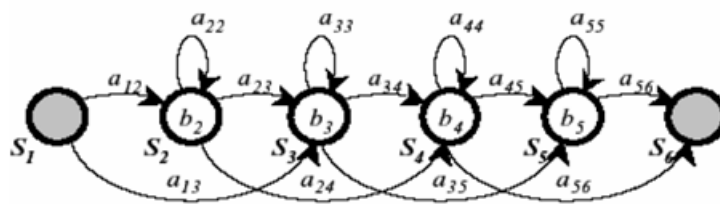


Figura 13: Topología básica de los HMM [40]

La Figura 13, nos indica la topología básica de los HMM, es la misma con la que hemos trabajado para nuestros modelos para cada una de las palabras.

“El modelo en sí tiene 4 estados ‘activos’, S_2, S_3, S_4, S_5 : El primer y el último estado (S_1 y S_6), son estados no ‘emisores’ (sin función de observación), usados por HTK para algunas facilidades de implementación.

Las funciones de observación b_i son distribuciones gaussianas con matrices diagonales. Las transiciones de probabilidades son nombradas

α_{ij} . En HTK un HMM se describe en un archivo de texto siguiendo una estructura definida . " [25]

4.2.5 Entrenamiento de los modelos

El proceso de entrenamiento de los modelos consta de ciertos pasos, que representa a cada palabra.

En sí depende de dos etapas:

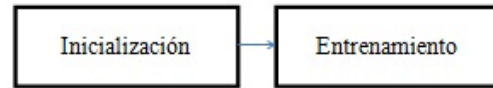


Figura 14: Proceso de entrenamiento HTK

Entrenamiento, Inicialización

Dentro de este proceso se realizan pasos para “inicializar” los valores de nuestros modelos, ya que cuando se definen los modelos ocultos de Markov para cada palabra, se impusieron o definieron variables con media cero y varianza uno.

Y estos valores van a ser modificados tomando en cuenta los mfcc.

HTK tiene dos herramientas para realizar esta tarea: HInit y HCompV.

HInit realiza una inicialización de los modelos por alineación en el tiempo de los datos de entrenamiento con el algoritmo de Viterbi.

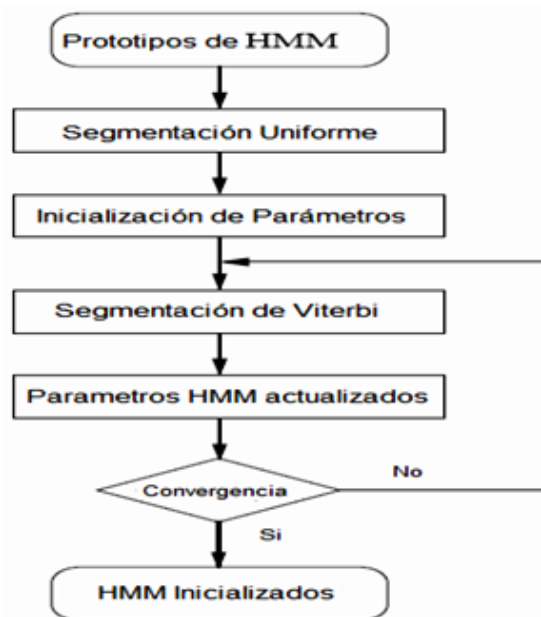


Figura 15: Operación de HInit

Mientras que la herramienta HCompV realiza una inicialización “absoluta” de los HMMs; esta herramienta genera la media y vectores de varianza calculados sobre todo el corpus.

En nuestro caso hemos utilizado la herramienta HInit y ha funcionado correctamente para nuestro prototipo.

Este proceso se debe ejecutar mediante la herramienta HInit y para cada uno de los modelos creados.

La manera de uso de cada herramienta HTK se lo puede encontrar en Libro HTK.

Entrenamiento, Re-estimación

En la fase de entrenamiento se debe re-estimar e iterar los valores principales de los HMM, con su inicialización previa.

Este proceso se puede decir que es el entrenamiento de los modelos tomando en cuenta su inicialización y los vectores de coeficientes cepstrales en la frecuencia de mel, generados inicialmente.

Este proceso HTK realiza con la herramienta HRest y se lo puede realizar las veces que se desee, obviamente utilizando la salida la Re-estimación N-1 como entrada para la Re-estimación N.

El proceso de re-estimación se lo puede explicar básicamente como un proceso que se encarga de las probabilidades y variación de parámetros de los HMM definidos.

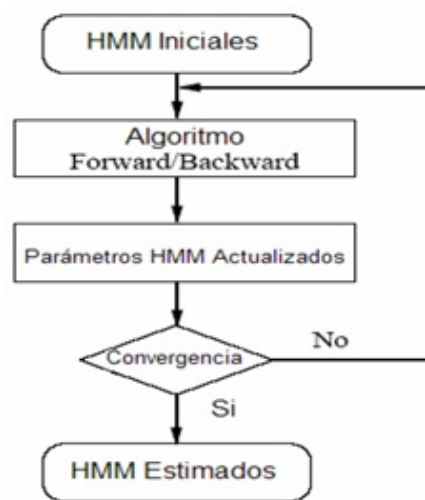


Figura 16: Diagrama de flujo, Re-estimación de los HMM [25]

4.2.6 Diccionario y Gramática

En este proceso se debe establecer un estándar para que el sistema reconozca el modo con el que va a trabajar y realizar el reconocimiento de palabras.

De nosotros va a depender el modo de operación de nuestro sistema, es decir, podemos establecer parámetros que permitan definir si nos va a reconocer: una, dos, tres o N palabras y si tenemos silencios iniciales y finales, entre palabras o cadena de palabras.

El diccionario y la gramática se crea de una manera sencilla con cualquier editor de texto simple, dentro de estos archivos se va a definir lo anteriormente mencionado.

La gramática para nuestro prototipo se ha establecido de la siguiente manera:

silencio inicial [sil] seguido de una sola palabra [PALABRA] y seguido de un silencio final [sil]

Este archivo se lo desarrolla mediante un lenguaje llamado PERL con el cual se pueden definir variables, y mediante estas se establece un estructura para el sistema.

De este modo el sistema debe reconocer que palabra corresponde a cada una de estas variables.

Compilación de la Gramática

HTK también facilita una herramienta para realizar una compilación de nuestra gramática, la herramienta utilizada es HParse, que genera un archivo con extensión “slf” que equivale a la red gramatical, el objetivo de la cual es establecer las especificaciones descritas en la gramática.

Dado que nuestra gramática es simple, pues tenemos el objetivo de reconocer un solo comando de voz por cada orden, entonces nuestra red quedaría definida como se indica en la figura 17.

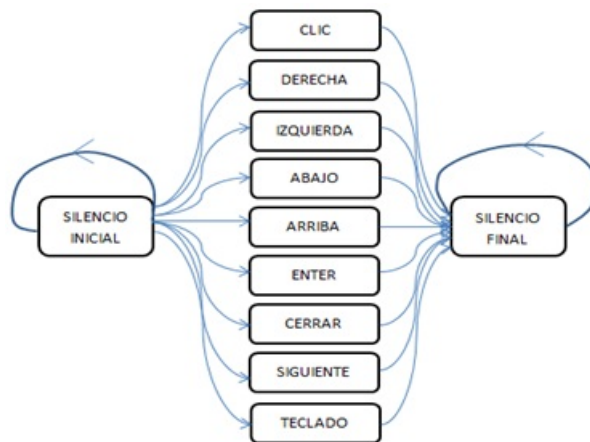


Figura 17: Red gramatical utilizada

En la figura 17 podemos observar la función básica con la que trabaja nuestro sistema, es decir, un estado inicial y final que sería nuestro modelo de silencio, y nuestra palabra central o comando de voz que podría ser cualquiera de las variables descritas (una sola palabra). De esta forma, nos queda únicamente realizar las pruebas de reconocimiento mediante todos nuestros archivos obtenidos en el proceso HTK.

4.3 PRUEBAS CON LA HERRAMIENTA SELECCIONADA

Luego de concluir con las fases de entrenamiento y diseño de nuestro sistema debemos realizar las pruebas para establecer un nivel de eficacia en el reconocimiento. Para ello, HTK también brinda la posibilidad de trabajar con herramientas para lo que son las pruebas, evaluaciones o análisis de resultados de nuestros modelos entrenados con algún tipo de entrada.

HTK es un toolkit que brinda muchas facilidades y entre ellas se encuentra la ventaja de probar nuestros modelos luego de la fase de entrenamiento.

La herramienta que ayuda en este proceso es HVite, que se encuentra descrita en el capítulo de “Adaptación HMM” dentro del libro de HTK.

Podríamos decir que lo único que hace es simular el proceso del algoritmo de Viterbi que tiene como función principal encontrar las secuencias de estados más probables en cada uno de nuestros HMM.

Lo que se hace es comparar los modelos entrenados con una señal de entrada que puede ser un archivo de voz pregrabado (.wav o .sig) o mediante la voz directa desde nuestro micrófono.

Luego de la comparación la herramienta va a imprimir como resultado las etiquetas que pertenecen a la palabra reconocida, permitiéndonos verificar la efectividad de reconocimiento de nuestro sistema.

El nivel de reconocimiento que presenta HTK es aceptable, basado en las pruebas de reconocimiento en vivo realizadas, podríamos definir un porcentaje de efectividad entre un 85 a 90 % y este podría aumentar dependiendo de parámetros que se especifican en el diseño del prototipo.

A continuación se indica la línea de código que hemos definido para probar los modelos entrenados de nuestras palabras:

```
HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H
modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_abajo -H
modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -
H modelo/hmm1/hmm_siguiente -H modelo/hmm1/hmm_clic -H
modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H mo-
delo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H mode-
lo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf
definicion/diccionario.txt definicion/listahmm.txt
```

- -C → indica la ubicación y el nombre del archivo de configuración.
- -g → indica al programa que reproduzca por los altavoces la señal de entrada.
- -H → indica que a continuación se van a especificar los nombres de los modelos de cada una de las palabras entrenadas.
- -w → define el nombre de la red gramatical. (Diccionario y gramática).
- listahmm.txt → archivo que contiene la lista de todos los nombres de los modelos.

En las Figuras 18 19 20 podemos observar la prueba realizada mediante la herramienta HVite, de HTK.

En el Figura 18, se establece toda la línea y se ejecuta sin ningún error, dejando el sistema en estado pendiente, para dar inicio a nuestras pruebas. Para indicar que se enviará una señal de prueba mediante el micrófono, presionamos la tecla “enter”, pronunciamos una de las palabras y nuevamente presionamos “enter” para indicar que hemos concluido con la grabación de la muestra de prueba. Inmediatamente nos dará el resultado de reconocimiento como se indica en la Figura 19, y podemos proseguir con el mismo proceso de la misma manera con diferentes palabras de nuestro vocabulario, como se puede apreciar en la Figura 20.

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

x izquierda == [152 frames] -71.0189 [Ac=-10794.9 LM=0.0] (Act=23.8)

READY[7]>
Press return to start sampling

x izquierda == [159 frames] -70.7455 [Ac=-11248.5 LM=0.0] (Act=23.8)

READY[8]>
Press return to start sampling
^C
willian@willian ~/Escritorio/willian $ HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_abajo -H modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -H modelo/hmm1/hmm_siguiete -H modelo/hmm1/hmm_clic -H modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H modelo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf definicion/diccionario.txt definicion/listahmm.txt

READY[1]>
Press return to start sampling

```

Figura 18: HVite en proceso

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

x izquierda == [159 frames] -70.7455 [Ac=-11248.5 LM=0.0] (Act=23.8)

READY[8]>
Press return to start sampling
^C
willian@willian ~/Escritorio/willian $ HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_abajo -H modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -H modelo/hmm1/hmm_siguiete -H modelo/hmm1/hmm_clic -H modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H modelo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf definicion/diccionario.txt definicion/listahmm.txt

READY[1]>
Press return to start sampling

x izquierda x == [159 frames] -67.0844 [Ac=-10666.4 LM=0.0] (Act=23.8)

READY[2]>
Press return to start sampling

```

Figura 19: Reconocimiento realizado

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda

^C
willian@willian ~/Escritorio/willian $ HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_abajo -H modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -H modelo/hmm1/hmm_siguiete -H modelo/hmm1/hmm_clic -H modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H modelo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf definicion/diccionario.txt definicion/listahmm.txt

READY[1]>
Press return to start sampling

x izquierda x == [159 frames] -67.0844 [Ac=-10666.4 LM=0.0] (Act=23.8)

READY[2]>
Press return to start sampling

x x x abajo x == [178 frames] -62.0132 [Ac=-11038.3 LM=0.0] (Act=23.8)

READY[3]>
Press return to start sampling

```

Figura 20: Pruebas continuas de reconocimiento

4.4 DISEÑO DEL PROTOTIPO

En esta sección vamos a detallar la manera exacta del desarrollo de nuestro sistema, de acuerdo al diseño del plan de experimentación se explicará de forma práctica lo que se hace en cada uno de los pasos de proceso.

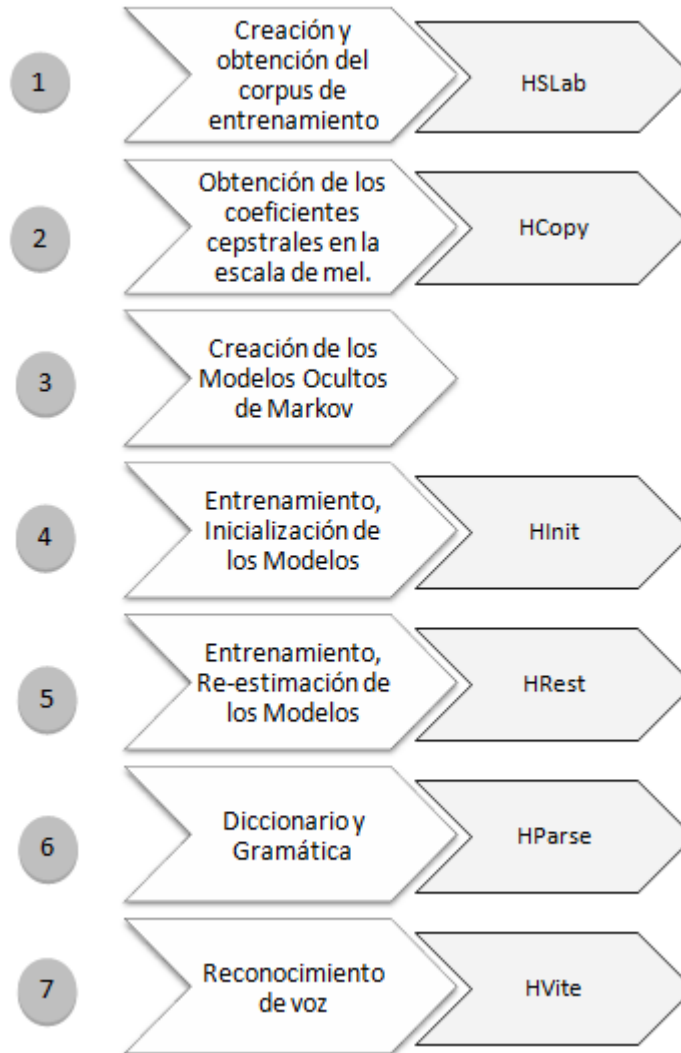


Figura 21: Pasos para el entrenamiento con HTK y sus herramientas

En la figura 21 podemos observar que se encuentran, los mismos 5 pasos primeros que se detallaron en la figura 10, pero esta vez tenemos dos pasos adicionales, el paso 6 y 7 se desarrollan para realizar pruebas de reconocimiento de voz mediante HTK, en nuestro caso en el paso 5 obtenemos lo que deseamos que son los modelos entrenados, y con estos posteriormente podemos acoplarlos a Julius, que es la siguiente fase del prototipo.

Herramientas utilizadas de HTK

Para hacer un uso adecuado de HTK vamos a mencionar las herramientas con las cuales se trabajará, ya que la instalación de HTK nos

proporciona varios procesos con distintos tipos de funcionalidad sobre los MOM, y obviamente, debemos emplear las que sean necesarias de acuerdo a nuestra aplicación.

HSLab: De las utilizadas es la única herramienta con interfaz gráfica, esta nos sirve para poder realizar las grabaciones de nuestras muestras de voz para el corpus de entrenamiento. También nos sirve para etiquetar la señal de nuestras grabaciones, esta herramienta se utiliza como paso inicial, pues gracias a ella obtenemos nuestra base de datos de entrenamiento.

HCopy: La utilizamos principalmente para la conversión de formatos de características y archivos, en nuestro caso utilizamos esta herramienta para convertir las muestras de voz tomadas inicialmente en formato wav o sig (formato htk) a vectores de coeficientes cepstrales en la frecuencia de mel, un tipo de formato mfcc.

HInit: Esta herramienta nos brinda la función de inicializar nuestros modelos tomando en cuenta los archivos mfcc creados mediante nuestras muestras de voz inicialmente con la herramienta HCopy.

HRest: Nos ayuda a re-estimar valores de nuestros modelos, es decir, plantear una etapa de entrenamiento sobre nuestros modelos inicializados, creando y modificando sus valores.

HParse: Herramienta HTK que nos ayuda a compilar una gramática creada por nosotros, y genera una red con su resultado (archivo.slf). evaluación ya sea mediante archivos de entrada pregrabados o directamente desde el micrófono.

4.4.1 Creación y obtención del corpus de entrenamiento

Obtener un corpus de entrenamiento es uno de los pasos más importantes y además del paso inicial para proceder al entrenamiento de nuestro prototipo.

Archivo de configuración

El archivo de configuración es un archivo de texto que contiene parámetros que establecen el modo de configuración y trabajo de las herramientas según se requiera.

Creación del archivo de configuración

Antes de proceder a explicar la fase de grabación de nuestro corpus de entrenamiento se va a detallar un paso importante para fases posteriores.

Se trata de un archivo de configuración que se envía como parámetro en la utilización de ciertas herramientas HTK.

Este archivo se puede crear con cualquier editor de texto, se debe guardar con extensión “.conf”.

A continuación vamos a indicar el archivo de configuración utilizado para nuestro sistema, el cual se llama “análisis.conf” [33]:

SOURCEFORMAT =HTK	#	Formato de los archivos de voz
TARGETKIND = MFCC_o_D_A	#	Coeficientes a usar
WINDOWSIZE = 200000.0	#	20 mseg como longitud de trama
TARGETRATE = 100000.0	#	10 mseg como periodicidad entre tramas
NUMCEPS = 12	#	Número de coeficientes MFCC
USEHAMMING = T	#	Uso de la ventana de Hamming
PREEMCOEF = 0.97	#	Coeficiente de pre-énfasis
NUMCHANS = 25	#	Número de canales del banco de filtros
CEPLIFTER = 22	#	Longitud del liftering en cepstrum

Figura 22: Archivo de configuración HTK

- Con SOURCEFORMAT = HTK Indicamos que hemos usado el formato propio de HTK para grabar las secuencias de voz.
- TARGETKIND = MFCC_o_D_A indica que vamos a extraer los coeficientes MFCC. (MFCC_o) coeficientes MFCC, (_D) coeficientes delta y (_A) coeficientes de aceleración. Esto significa que los coeficientes delta y los coeficientes de aceleración se calculan y se anexa a los coeficientes MFCC estáticos obtenidos [40].
- NUMCEPS = 12 indica que se calcularán los 12 coeficientes MFCC, del total 39 coeficientes.
- El tamaño de la ventana será de 25 mseg , indicado por WINDOWSIZE = 250000.0, y se tomará una trama cada 10 mseg , por lo que habrá solapamiento (TARGETRATE = 100000.0). Son el resultado de (MFCC_o = 13), además de los coeficientes delta (+13) y de los coeficientes de aceleración (+13).
- USEHAMMING = T activa el uso de la ventana de Hamming. Se usará un coeficiente de preénfasis de valor $\alpha = 0,97$ (PREEMCOEF = 0.97).
- El filtro perceptual $H_m(k)$ que participa en el cálculo de los MFCC está dividido en 25 bandas (NUMCHANS = 25) y CEPLIFTER = 22 indica el número de muestras del cepstrum de la voz que se recortan.

La señal fue segmentada en paquetes sucesivos, cuya longitud es de 20ms. Cada paquete fue multiplicado por la función de Hamming utilizando ventanas de una misma duración.

Se extrajo un paquete de vectores de coeficientes acústicos, dando una representación compacta de las propiedades espectrales del paquete.

En este trabajo se utilizó el análisis MFCC o análisis de los Coeficientes Cepstrales en la escala de Frecuencia de Mel. Se obtuvieron los primeros 12 coeficientes MFCC $[c_1, \dots, c_{12}]$, el coeficiente MFCC “nulo” (c_0) el cual es proporcional al logaritmo de la energía total, 13 “coeficientes delta” y 13 “coeficientes de aceleración”

Todos los coeficientes en conjunto forman un vector de 39 coeficientes extraídos de cada señal de voz. El banco de filtros utilizados es de 26 canales. Las señales tuvieron un preénfasis de primer orden con un coeficiente de 0.97 [25].

Ventana de Hamming

Las funciones ventana son funciones matemáticas utilizadas generalmente en el análisis y procesamiento de señales, que nos permite aislar

una porción de la señal a analizar y desechar el resto en mayor o menor medida. Existen muchos tipos de ventanas para trabajar con el procesamiento de señales, que permiten obtener distintos resultados en el dominio de las frecuencias y una de ellas es la ventana de Hamming [22].

Coefficiente de preénfasis

El preénfasis es el incremento del nivel de altas frecuencias de audio en proporción directa al aumento de amplitud del ruido en dichas frecuencias, antes de la modulación, con el fin de mantener una relación constante a través de toda la banda de transmisión [8].

El filtro perceptual

Es un sistema de compresión con pérdida, esto quiere decir que el sonido original y el comprimido no son exactamente iguales. Estas pérdidas tienen en cuenta el funcionamiento del oído humano, de tal forma que aunque los sonidos no son iguales se perciben como si lo fuesen [5].

Topología

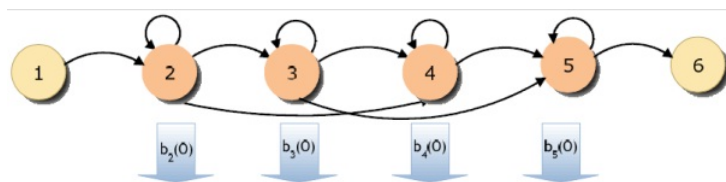


Figura 23: Topología Usada HMM [33]

Descripción [33]:

- “Número de estados: entre 4 y 8 estados activos, pues trataremos con palabras cortas. Además, se añaden un primer y último estado que no generan observación por motivos de implementación en HTK.
- Forma de la función de observación asociada a cada estado: $b_j(k)$ son funciones densidades de probabilidad gaussianas unidimensionales (con matrices diagonales de 39 elementos no nulos), inicializadas con media cero y varianza uno. Cada elemento de la diagonal representa a un coeficiente MFCC.
- Transiciones entre estados: una matriz cuadrada, de dimensión igual al número de estados los a_{ij} deben implementar la topología tipo Bakis de la figura anterior. De cada estado se puede volver a él mismo, o transitar a los dos estados siguientes (excepto el primer y último estado, que son transitorios).”

Como podemos observar, se establecen parámetros de configuración que detallan valores con los que se va ejecutar la herramienta HTK. Más detalles de los parámetros de este archivo podemos encontrarlos en [40].

Proceso de Grabación

Como mencionamos en el plan de experimentación, lo primero que se debe hacer es obtener nuestra base de datos de entrenamiento o corpus. Para ello podemos utilizar una herramienta que nos brinda HTK, llamada HSLab.

La sintaxis de utilización de esta herramienta es muy parecida a todas las herramientas HTK, es decir, se hace la llamada al nombre de la herramienta, seguida de los parámetros necesarios y operación a realizar.

Las palabras a grabar para nuestro prototipo son las siguientes: Si-guiente, derecha, izquierda, abajo, arriba, enter, cerrar, teclado, gracias.

Para cada una de estas palabras debemos hacer una cantidad N de grabaciones, ahora de esto va a depender mucho la eficacia de reconocimiento como en cualquier método de entrenamiento, mientras mayor número de muestras exista en nuestro corpus de entrenamiento, mayor será el porcentaje de efectividad de reconocimiento.

Otro factor del cual va a depender la efectividad de reconocimiento, es el locutor que realice las grabaciones. Por ello, si todas las grabaciones han sido realizadas por el mismo locutor, el sistema va a tener un mayor grado de reconocimiento con esa persona.

Esto no quiere decir que con una persona diferente el sistema no pueda realizar el reconocimiento, pero quizá se presente un porcentaje mayor de error en este proceso.

Se puede realizar el sistema de la manera que se desee; tomando como base de entrenamiento la voz de una sola persona a la cual las pruebas van a ser mucho más efectivas, o tomando muestras de voz de diferentes personas de las palabras indicadas anteriormente, para que de este modo disminuya el porcentaje de error cuando se haga uso del sistema por distintas personas.

Para invocar HSLab procedemos de la siguiente forma:

HSLab *nombre.sig*

Podemos observar que es una manera sencilla, aunque también se pueden especificar ciertas características para el archivo de audio que se va a grabar.

HSLab -C *archivo.conf* *nombre.sig*

Donde:

- - C *archivo.conf* → -C, este parámetro indica que se va a utilizar ciertas configuraciones para la grabación, seguidamente viene el nombre del archivo de configuración.
 - *nombre.sig* → este parámetro simplemente nos indica el nombre con el que va a guardar nuestra muestra y consecuentemente el nombre de las etiquetas realizadas en nuestros archivo de voz.
- Una vez ejecutada la herramienta, se presentará una interfaz gráfica de la siguiente manera:

Procedemos a grabar cada una de las palabras y las veces que se hayan establecido, mediante el botón “rec” y para terminar la grabación

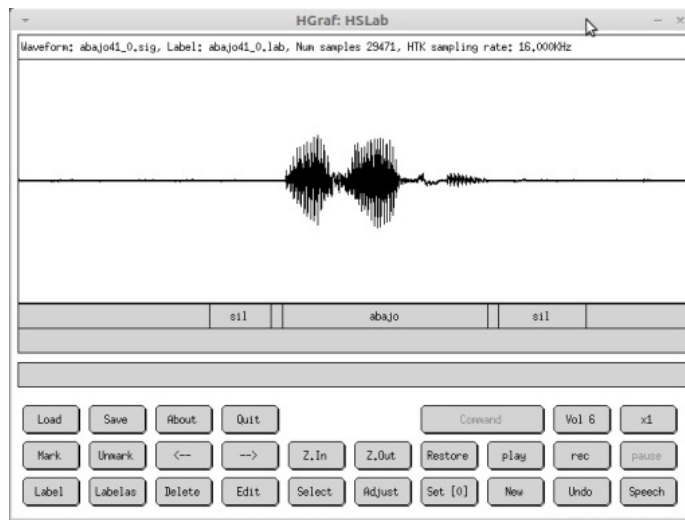


Figura 24: HSLab Ejemplo Etiquetado

“stop”. Mediante “play” podemos verificar si nuestra grabación a sido realizada correctamente como lo deseábamos.

Una vez terminada la grabación vamos a observar un gráfico de la onda de nuestra señal de voz, como se observa en la Figura 24. Posteriormente debemos, etiquetar nuestra señal, tarea que puede realizarse sobre el gráfico de nuestra grabación. Las etiquetas se las realizan mediante el botón “Labelas”.

En primer lugar seleccionamos la zona que vamos a etiquetar, obviamente la zona del gráfico en la que existen los puntos mas altos y bajos es la palabra pronunciada, entonces seleccionamos con el puntero del mouse esta zona y damos clic en “Labelas” y escribimos el nombre que se desee definir para describir a la palabra o zona.

Nosotros utilizamos los mismos nombres de las palabras para las etiquetas de cada una de ellas. También se debe definir etiquetas para un silencio inicial y silencio final entre la palabra grabada.

La etiqueta de silencio debe ser la misma para todas las palabras (en nuestro caso se ha definido con el nombre de “sil”). Ahora procedemos a guardar las etiquetas con la opción “Save”, nos dará la posibilidad de cambiar el nombre de las etiquetas de cada grabación que se guardará en un archivo de extensión “.lab”, recomendamos dejar el nombre estándar pues de esta manera podremos saber a qué archivo de voz (.sig) pertenece cada archivo de etiquetas (.lab). Seguido del nombre, presionamos la tecla entrar, para aceptar la opción de guardado.

Y en ese momento esto nos generará dos archivos: uno que es nuestro archivo de audio “nombre.sig” y nuestro archivo de etiquetas “nombre.lab”, los cuales nos servirán como paso inicial y básico para procesos posteriores.

4.4.2 Obtención de los Coeficientes Cepstrales en la escala de Mel

Este proceso de análisis es el primer paso que se realiza luego de tener ya nuestro corpus para trabajar. Esto se refiere a que debemos convertir nuestros archivos de audio “.sig” a vectores de coeficientes para poder procesar sus características.

Como se mencionó en el diseño del plan de experimentación, debemos realizarlo mediante la herramienta **HCopy**.

Esta herramienta al igual que HSLab, recibe como parámetro en archivo de configuración que sirve para realizar la conversión basándose en los parámetros descritos [25].

La manera de usarla es la siguiente:

HCopy -C archivo.conf -S lista.txt

- -S lista.txt → lista.txt es un archivo de texto común, que contiene dos columnas.

Archivo “lista.txt”

datos/sig/aw0.sig	datos/mfcc/aw0.sig
datos/sig/aw1.sig	datos/mfcc/aw1.sig
datos/sig/aw2.sig	datos/mfcc/aw2.sig
datos/sig/aw3.sig	datos/mfcc/aw3.sig
datos/sig/aw4.sig	datos/mfcc/aw4.sig
datos/sig/aw5.sig	datos/mfcc/aw5.sig
datos/sig/aw6.sig	datos/mfcc/aw6.sig
datos/sig/aw7.sig	datos/mfcc/aw7.sig
datos/sig/aw8.sig	datos/mfcc/aw8.sig
.....	

Figura 25: Archivo lista para herramienta HCopy

Este archivo debe contener todos los datos de entrenamiento.

La primera columna indica la ubicación y el nombre de cada uno de nuestros archivos de audio “.sig”, un espacio en blanco y seguido la segunda columna, que indica la ubicación donde se desea crear los archivos mfcc con su respectivo nombre, se recomienda colocar el mismo nombre para distinguirlos, simplemente cambiando la extensión.

Si esta todo correcto, nos va a generar un archivo mfcc por cada grabación .sig, en nuestro caso dentro del directorio “datos/mfcc/”.

4.4.3 Creación de los HMM

Para realizar el siguiente paso debemos crear nuestros modelos para cada una de nuestras palabras [33].

En el plan de experimentación definimos la topología con la que se va a trabajar.

Vamos a indicar la manera en la que hemos creado nuestros modelos, especificando un ejemplo estándar, el cual vamos a utilizar para todas las palabras.

Descripción del archivo que define a los MOM [33]:

```

~o <VecSize> 39 <MFCC_o_D_A>
~h "clie"
<BeginHMM>
<NumStates> 6
<State> 2
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 4
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 5
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<TransP> 6
0.0 0.5 0.5 0.0 0.0 0.0
0.0 0.4 0.3 0.3 0.0 0.0
0.0 0.0 0.4 0.3 0.3 0.0
0.0 0.0 0.0 0.4 0.3 0.3
0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>

```

Figura 26: Ejemplo de definición de un HMM

- `o<VecSize>39<MFCC_o_D_A>` → nos indica la longitud del vector de coeficientes y el tipo de coeficientes.
- `~h"clit" < BeginHMM > (...) < EndHMM >` → contiene toda la definición del HMM llamado "clit".
- `<NumStates>6` → nos indica el número total de estados, cuatro activos y dos transitorios.
- `<State>i` → "introduce a la especificación de las características del estado i-ésimo. La matriz diagonal que representa la probabilidad de generación en ese estado puede especificarse poniendo simplemente los elementos de la diagonal: un vector de medias y otro de varianzas para las 39 gaussianas. Los estados 1 y 6 no se describen, pues no tienen función de observación."
- `<Mean> 39` → define el vector de medias de 39 elementos de la función de observación del estado al que acompaña. Estos coeficientes serán entrenados posteriormente y sus valores cambiarán, pero inicialmente son definidos en cero.
- `<Variance> 39` especifica el vector de varianzas de 39 elementos de la función de observación del estado al que acompaña. Son inicializados a uno, aunque serán entrenados posteriormente.
- `<TransP> N` especifica la matriz de transición de estados, de dimensiones $N \times N$ [33].

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$$

- Los valores nulos indican transiciones no permitidas entre estados. El resto de valores pueden ser inicializados arbitrariamente, pero los elementos de cada línea deben sumar 1. Se recomienda en lo posible, que todos los elementos no nulos tengan el mismo valor, para que ninguna transición tenga una ventaja inicial en el entrenamiento.
- Este prototipo debe copiarse para cada palabra que hemos entrenado, simplemente cambiando el nombre del archivo (`hmm_clit`) y la cabecera (`~h"clit"`).

4.4.4 Inicialización de Modelos

El siguiente paso que se debe realizar es la inicialización de los modelos creados [25].

Para este proceso hemos utilizado la herramienta `HInit`, ya que obtuvimos buenos resultados para nuestro sistema.

Esta herramienta se usa de la misma manera, es decir, desde el terminal priorizando su nombre y los parámetros necesarios, como se indica a continuación:

HInit -S ruta/entrenamiento.txt -M ruta_salida -H ruta_modelos/hmm_nombre -l nombre_etiqueta -L ruta_etiquetas nombre_hmm

- -S entrenamiento.txt → se trata de un archivo que contiene una lista de las ubicaciones de nuestros archivos mfcc.
- -M ruta_salida → indica el directorio en el cual se van a crear los modelos inicializados.
- -H ruta_modelos → indica el directorio o ruta en la que se encuentra el modelo a inicializar, seguido de su nombre.
- -l nombre_etiqueta → indica la etiqueta de la palabra que representa nuestro modelo que se esta inicializando.
- -L ruta_etiquetas → indica el directorio en el cual se encuentran nuestras etiquetas, en nuestro caso es "datos/lab/" .
- nombre_hmm → se especifica el nombre que se definió dentro de nuestro modelo "~h"derecha"".

Una vez que hemos especificado el formato de uso de la herramienta HInit, ahora vamos a detallar como lo hemos utilizado en nuestro prototipo.

HInit -S entrenamiento/entrenamiento.txt -M modelo/hmmo -H modelo/proto/hmm_derecha -l derecha -L datos/lab/ derecha

La línea anterior esta inicializando nuestro modelo para la palabra "derecha".

A continuación se especifican sus parámetros:

- -S entrenamiento.txt → archivo que contiene una lista de las ubicaciones de nuestros archivos mfcc.
- -M → indica el directorio en el cual se van a crear los modelos inicializados, en nuestro caso los vamos a guardar en "modelo/hmmo".
- -H → indica el nombre archivo del modelo a inicializar, especificando la ruta en la que se encuentra tal archivo, en nuestro caso los modelos creados para cada una de las palabras se encuentran en "modelo/proto/" y a continuación deberíamos detallar el nombre de nuestro modelo "modelo/proto/hmm_derecha".
- -l derecha → indica la etiqueta de la palabra que representa nuestro modelo, es decir, cómo definimos nuestra etiqueta en el momento de nuestra grabación, esta información se encuentra en nuestros archivos ".lab".
- -L datos/lab/ → indica el directorio en el cual se encuentran nuestras etiquetas, en nuestro caso es "datos/lab/" .
- derecha → es el nombre del modelo que vamos a inicializar (~h"derecha").

```

datos/mfcc/aw0.mfcc
datos/mfcc/aw1.mfcc
datos/mfcc/aw2.mfcc
datos/mfcc/aw3.mfcc
datos/mfcc/aw4.mfcc
datos/mfcc/aw5.mfcc
datos/mfcc/aw6.mfcc
datos/mfcc/aw7.mfcc
datos/mfcc/aw8.mfcc
.....

```

Figura 27: Contenido de archivo "entrenamiento.txt"

Contenido del archivo "entrenamiento.txt"

El proceso anterior debemos realizarlo para cada uno de nuestros modelos, uno a uno, especificando cada parámetro de acuerdo a la palabra que se esté tratando.

No debemos olvidar que el silencio también tiene su propio modelo, por lo que se lo va a tratar como si fuera una palabra más.

Una vez que corremos la línea de inicialización para cada uno de los modelos, debemos fijarnos que no nos presente ningún error.

Luego de finalizado este proceso, encontraremos en el directorio de salida nuevos archivos con el nombre especificado, estos serían nuestros modelos inicializados, pues podríamos observar que su contenido ya no es el mismo que el definido inicialmente.

4.4.5 Re-estimación de los modelos

Este proceso es el encargado de realizar una re-estimación de los modelos que hemos obtenido luego de la inicialización.

Como mencionamos en el plan de experimentación, este proceso se lo realiza con una de las herramientas que nos brinda HTK, que es HRest.

Su formato estándar de uso es:

HRest -S ruta/entrenamiento.txt -M ruta_salida -v 2e-3 -H ruta/hmm_nombre -l etiqueta -L ruta_etiquetas nombre_modelo

- -S ruta/entrenamiento.txt → es el mismo parámetro que se utilizó para la herramienta HInit.
- -M ruta_salida → es el directorio en el que se va a ubicar nuestro modelo entrenado.
- -v 2e-3 → es el valor de varianza con el que hemos trabajado, un valor estándar.
- -H ruta/hmm_nombre → al igual que en el paso de inicialización, debemos especificar la ruta de nuestros archivos inicializados, seguidos del nombre del modelo inicializado.
- -l etiqueta → nombre de la etiqueta
- -L rutas_etiqueta → directorio o ubicación de nuestros archivos de etiquetas.

- nombre_modelo → especifica el nombre del modelo (~h "derecha").

A continuación se especifica la manera que trabajamos en nuestro sistema para el entrenamiento del modelo “derecha”:

HRest -S entrenamiento/entrenamiento.txt -M modelo/hmm1 -v 2e-3 -H modelo/hmmo/hmm_derecha -l derecha -L datos/lab/ derecha

- -S → especificamos el directorio y el nombre del archivo de entrenamiento, que es el mismo que hemos utilizado en el proceso de inicialización.
- -M modelo/hmm1 → nuestro directorio de salida, en el cual se van a ubicar los modelos entrenados, en nuestro caso se crearán los modelos entrenados en “modelo/hmm1”.
- -v2e-3 → es la varianza que utilizamos.
- -H modelo/hmmo/hmm_derecha → especificamos la ruta de nuestro modelo “modelo/hmmo/” y su nombre “hmm_derecha”.
- -l derecha → es la etiqueta de nuestra palabra, modelo.
- -L datos/lab/ → se especifica el directorio o ubicación de los archivos de etiquetas (archivos “.lab”).
- derecha → es el nombre de nuestro modelo.

Una vez que ejecutamos la línea especificada, y si esta correcto todo y no se presenta ningún error, deben crearse los archivos entrenados en el directorio “modelo/hmm1/” (obtenemos un archivo nuevo por cada modelo con el mismo nombre).

Los archivos obtenidos con este último paso son de suma importancia, ya que son los archivos principales que contienen nuestros modelos entrenados.

Ahora, para establecer una forma de prueba para reconocimiento podemos aprovechar que HTK nos brinda la posibilidad de trabajar con herramientas de prueba y reconocimiento. Para ello debemos realizar pasos adicionales, que se describen a continuación.

4.4.6 Diccionario y Gramática

Para poder establecer un formato de reconocimiento para nuestro sistema debemos definir un par de archivos simples de texto, para que nos ayuden en este proceso.

Gramática

La gramática define la estructura de reconocimiento del sistema, a continuación se indica la estructura utilizada para nuestro sistema.

Contenido del archivo gramatica.txt

```
$funcion = derecha | izquierda | abajo | arriba | teclado |
enter | gracias | siguiente | clic | cerrar;
( {sil} [ $funcion ] {sil} )
```

Figura 28: Gramática HTK utilizada

Su definición es simple, se lo realiza mediante el lenguaje Perl, de la forma que se indica en la Figura 28.

En primer lugar definimos una variable con los valores posibles que se nos puede presentar, el signo “\$” indica que estamos definiendo una variable que en nuestro caso se llama “funcion” y esta se iguala a sus posibles valores, es decir: “derecha”, “izquierda”, “abajo”, etc. separados con “|”, que indica cada una de las diferentes alternativas que puede tomar nuestra variable “funcion”, e indicamos que ha finalizado la orden con punto y coma “;”.

A continuación definimos una línea en donde establecemos la forma que el sistema va a reconocer.

Las llaves denotan cero o varias repeticiones, es decir “sil”, que representa al silencio (definido en el diccionario) se puede repetir una o más veces.

Los corchetes “[]” alrededor de “\$funcion” significa cero o alguna ocurrencia.

Y todo esto dentro de paréntesis, indicando que es una sola cadena.

Diccionario

El diccionario es un archivo de texto en el cual se definen las variables a utilizar, es decir, nuestros modelos principalmente. A continuación se describe el archivo utilizado para nuestro prototipo.

```
derecha [derecha] derecha
izquierda [izquierda] izquierda
abajo [abajo] abajo
arriba [arriba] arriba
enter [enter] enter
siguiente [siguiente] siguiente
gracias [gracias] gracias
teclado [teclado] teclado
clic [clic] clic
cerrar [cerrar] cerrar
sil [sil] sil
```

Figura 29: Diccionario HTK utilizado

En el archivo se definen tres columnas como podemos observar, la primera columna de izquierda a derecha, define los nombres de las variables en el archivo de gramática.

La columna la central es opcional, esta define a manera de etiquetas la palabra que va a aparecer en el momento que identifique la misma.

Y la tercera columna, es el nombre que se define en cada uno de los modelos con el símbolo “~h”.

Red Gramatical

La red gramatical es un tipo de archivo con extensión “slf” que se crea mediante una herramienta HTK, llamada HParse.

Para nuestro prototipo lo realizamos de la siguiente manera, simplemente con dos parámetros:

```
HParse definicion\gramatica.txt definicion\red.slf
```

Como se aprecia, enviamos como parámetro el archivo de gramática creado anteriormente, que en nuestro caso es “gramatica.txt” y un segundo parámetro que es el nombre “red.slf”.

En el momento que se ejecuta correctamente la línea, se creará el archivo “red.slf” en la ubicación que se definió, el cual lo utilizaremos cuando necesitemos hacer una prueba de reconocimiento mediante HTK.

4.4.7 Reconocimiento de voz

Mediante HTK podemos realizar un reconocimiento a manera de pruebas para verificar el reconocimiento de nuestras palabras entrenadas.

Para ello utilizamos una de las herramientas HTK, que lo que hace es simular el algoritmo de Viterbi (tratado en el capítulo 3).

La herramienta se llama HVite, de igual manera se la utiliza directamente desde el terminal, con los parámetros respectivos.

La herramienta HVite en nuestro caso se utiliza de la siguiente manera:

```
HVite -C analisis/analisis2.conf -g -H modelo/hmm1/hmm_s -H
modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_izquierda -
H modelo/hmm1/hmm_abajo -H modelo/hmm1/hmm_arriba -H
modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_cerrar -H mo-
delo/hmm1/hmm_gracias -H modelo/hmm1/hmm_siguiete -H
modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_enter -H mo-
delo/hmm1/hmm_inicio -H modelo/hmm1/hmm_clic -w definicion/red.slf
definicion/diccionario.txt definicion/listahmm.txt
```

- -C análisis/analisis2.conf → es el mismo archivo de configuración pero modificado.
- -g indica que reproduzca por los altavoces la señal de entrada o grabación.
- -H indica cada uno de los modelos que se van a comparar (modelos entrenados, obtenidos en el paso de entrenamiento), aquí es donde debemos incluir todos los modelos que queremos que reconozca en nuestra prueba.
- -w → en este parámetro debemos establecer el directorio y el nombre de nuestra red gramatical, que creamos con la herramienta HParse; luego de un espacio colocamos el nombre de nuestro diccionario y seguido de un nuevo espacio para finalmente colocar el nombre de una lista que no es mas que un archivo de texto con una lista con los nombres de todos nuestros modelos.

En el archivo análisis2.conf, es el mismo contenido de archivo “análisis.conf” pero agregamos las siguientes líneas, que nos servirán para usarlo como configuración para la prueba de reconocimiento.

```
SOURCERATE = 625.0      # 16 kHz de frecuencia de muestreo
SOURCEKIND = HAUDIO     # Entrada directa de audio
AUDIOSIG = -1           # Reconocimiento controlado por teclado
```

Con estos cambios nuestro archivo “análisis2.conf” quedaría de la siguiente manera:

SOURCEFORMAT = HTK	# Formato de los archivos de voz
TARGETKIND = MFCC_o_D_A	# Coeficiente a usar
WINDOWSIZE = 2000000.0	# 20 mseg como longitud de trama
TARGETRATE = 100000.0	# 10 mseg como periodicidad entre tramas
NUMCEPS = 12	# Numero de coeficientes MFCC
USEHAMMING = T	# Uso de la ventana de Hamming
PREEMCOEF = 0.97	# Coeficiente de pre-énfasis
NUMCHANS = 25	# Número de canales del banco de filtros
CEPLIFTER = 22	# Longitud del liftering en cepstrum
SOURCERATE = 625.0	# 16 kHz de frecuencia de muestreo
SOURCEKIND = HAUDIO	# Entrada directa de audio
AUDIOSIG = -1	# Reconocimiento controlado por teclado

Figura 30: Archivo de configuración HTK utilizado

Contenido del archivo “listahmm.txt”:

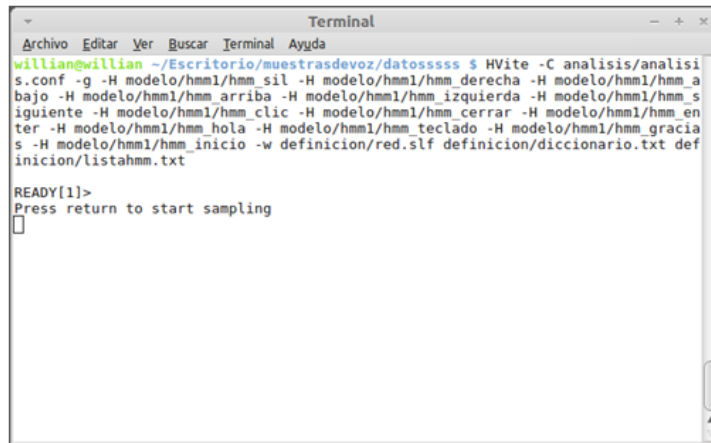
sil
derecha
izquierda
arriba
abajo
teclado
inicio
gracias
clic
enter
cerrar
siguiente

Figura 31: Listado de HMM utilizando HTK

Como podemos observar en la Figura 31 simplemente se trata de una lista de cada uno de los nombres de nuestros modelos o palabras entrenadas. Una vez que todos los archivos están listos para usarlos como parámetros en HVite, ejecutamos la línea explicada anteriormente y si todo esta correcto, tendremos el siguiente resultado.

La Figura 32 nos indica que el sistema está listo para recibir las muestras de voz de entrada para empezar el reconocimiento.

Entonces, presionamos la tecla “Enter” ,pronunciamos una de las palabras entrenadas, y nuevamente otro “Enter” para indicar que hemos terminado la grabación.



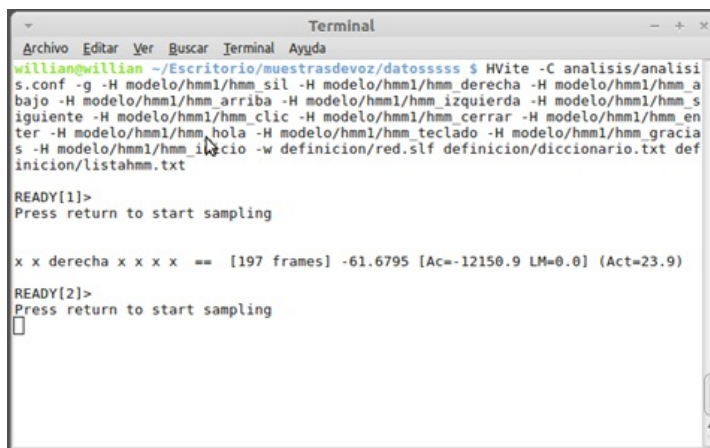
```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
willian@willian ~/Escritorio/muestrasdevoz/datosssss $ HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_a_bajo -H modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -H modelo/hmm1/hmm_siguiente -H modelo/hmm1/hmm_clic -H modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H modelo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf definicion/diccionario.txt definicion/listahmm.txt
READY[1]>
Press return to start sampling

```

Figura 32: Ejecución herramienta HVite

Seguidamente nos dará el resultado del reconocimiento, como se indica a continuación:



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
willian@willian ~/Escritorio/muestrasdevoz/datosssss $ HVite -C analisis/analisis.conf -g -H modelo/hmm1/hmm_sil -H modelo/hmm1/hmm_derecha -H modelo/hmm1/hmm_a_bajo -H modelo/hmm1/hmm_arriba -H modelo/hmm1/hmm_izquierda -H modelo/hmm1/hmm_siguiente -H modelo/hmm1/hmm_clic -H modelo/hmm1/hmm_cerrar -H modelo/hmm1/hmm_enter -H modelo/hmm1/hmm_hola -H modelo/hmm1/hmm_teclado -H modelo/hmm1/hmm_gracias -H modelo/hmm1/hmm_inicio -w definicion/red.slf definicion/diccionario.txt definicion/listahmm.txt
READY[1]>
Press return to start sampling

x x derecha x x x x == [197 frames] -61.6795 [Ac=-12150.9 LM=0.0] (Act=23.9)
READY[2]>
Press return to start sampling

```

Figura 33: Reconocimiento HVite

Como podemos observar, nos imprime la etiqueta de la palabra o modelo que ha reconocido, en este caso es la palabra "Derecha". De esta forma, podremos continuar realizando pruebas de reconocimiento, con todas las palabras entrenadas, de la misma manera que como se ha indicado.

4.4.8 Julius [33]

Para complementar el trabajo realizado con HTK, hemos utilizado otra herramienta de código abierto, la cual nos sirve específicamente para lo que es el reconocimiento de voz continua, LVCSR (Large Vocabulary Continuous Speech Recognition) [17].

Una de las ventajas de utilizar Julius, aparte de que nos brinde reconocimiento en tiempo real, es la compatibilidad que tiene con nuestra herramienta principal HTK, además nos brinda facilidades de adaptación para trabajar con los modelos entrenados obtenidos mediante HTK.

Lo que se hace es enviar los modelos entrenados como uno de los parámetros de Julius, y de esta manera se comparará la muestra de

entrada con cualquiera de ellos y en el momento de localizar la mayor semejanza con cualquiera de ellos, Julius imprimirá el resultado.

Proceso de acoplamiento

Para completar lo requerido en nuestro prototipo, hemos decidido trabajar con Julius, por lo que obviamente va a existir un proceso intermediario entre HTK y el mismo a fin de acoplar las características de cada una de ellas y obtener el resultado planteado para nuestro prototipo.

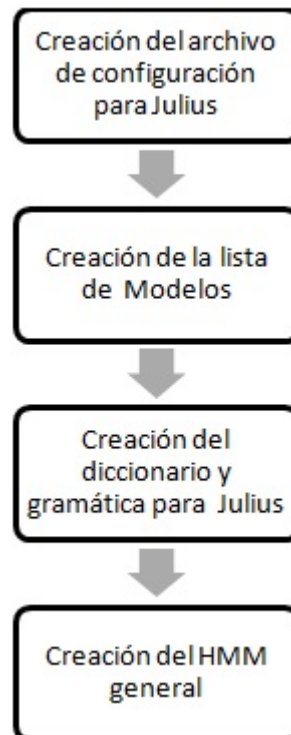


Figura 34: Pasos para el proceso de desarrollo en Julius

Creación del archivo de configuración

Archivo de configuración

Julius al igual que ciertas herramientas de HTK, necesita de un archivo de configuración, como uno de los parámetros principales que recibe, y podemos definir toda su configuración dentro de este archivo con extensión “.jconf”. Una vez que hemos obtenido los modelos entrenados mediante el proceso con HTK, podemos continuar con los siguientes pasos que son el desarrollo de la fase de Julius.

A continuación se indicará el archivo de configuración usado para nuestro prototipo:

Los dos últimos parámetros son simplemente para activar funciones de Julius.

```
-dfa wilo.dfa
-v wilo.dict
-h hmmdefs3
-hlist tiedlist
-demo
-input mic
```

Figura 35: Archivo de configuración Julius

- - demo → activa un reconocimiento continuo, evitando resultados con texto que no es de mucha utilidad, es decir, que imprima lo más importante, en nuestro caso las palabras reconocidas y la frase final de reconocimiento.
- - input mic → indica la manera de tomar las muestras de voz de entrada, en este caso “mic” quiere decir que la señal de entrada será tomada con el micrófono.
- - hlist → define el nombre de un archivo que contiene la lista de los MOM utilizados.
- - h → se define el nombre del archivo que contiene los MOM.
- - v → archivos de diccionario y gramática.
- - dfa → archivos de diccionario y gramática.

Creación de la lista de los modelos

De igual manera, se debe establecer dentro de uno de los parámetros una lista de los nombres de los modelos, este se define con el parámetro “hlist” de la siguiente manera:

- - hlist nombre del archivo

En nuestro caso sería:

-hlist tiedlist

Contenido del archivo “tiedlist”:

```
arriba
izquierda
gracias
teclado
clic
cerrar
enter
derecha
abajo
siguiente
sil
hola
inicio
```

Figura 36: Archivo “tiedlist”, listado de Modelos utilizados

Como podemos observar en la Figura 36, el archivo “tiedlist” únicamente contiene el listado de los modelos que se han entrenado con HTK y van a ser utilizados para el reconocimiento. A continuación se indicará

cómo se obtienen los demás parámetros, pues estos ya dependen del diseño del programador.

Diccionario y Gramática para Julius

En primer lugar, debemos crear dos archivos que equivalen a la gramática y al diccionario para Julius. Los dos archivos en nuestro caso se llaman: “wilo.voca” y “wilo.grammar” y deben tener el mismo nombre. Esto será necesario para un paso posterior.

Mediante estos dos archivos vamos a crear 3 archivos más, los cuales van a ser enviados como algunos de los parámetros indicados anteriormente en la sección 4.4.8.

Diccionario Julius

Contenido del archivo “wilo.voca”

% NS_B	
<s>	sil
% NS_E	
</s>	sil
% FUNCION	
derecha	derecha
abajo	abajo
izquierda	izquierda
arriba	arriba
siguiente	siguiente
clic	clic
cerrar	cerrar
enter	enter
gracias	gracias
teclado	teclado
hola	hola
inicio	inicio

Figura 37: Archivo .voca

El archivo de extensión “.voca” siempre tendrá la misma estructura, obviamente variando de acuerdo a los modelos en uso.

Explicación del archivo “.voca”:

El siguiente parámetro es para definir el silencio inicial, donde “sil” indica el nombre del modelo que representa el silencio.

```
%NS_B
<s>      sil
```

Para el silencio final es el mismo proceso, ya que se trata del mismo modelo de silencio, como observamos a continuación.

```
%NS_E
</s>     sil
```

El siguiente párrafo lo que describe es la variable principal que se va tomar en cuenta para el reconocimiento. Dicho párrafo consta de dos columnas: la de la izquierda es el nombre de la variable para Julius y la de la derecha es el nombre del modelo que representa, es decir, el

nombre de los modelos entrenados con HTK.

%FUNCION	
derecha	derecha
abajo	abajo
izquierda	izquierda
arriba	arriba
siguiente	siguiente
clic	clic
cerrar	cerrar
enter	enter
gracias	gracias
teclado	teclado
hola	hola
inicio	inicio

Figura 38: Función Para realizar el reconocimiento

Su contenido y estructura va a estar relacionado con el otro archivo “wilo.grammar”, por ejemplo, en la definición de las variables: NS_B, NS_E y FUNCION.

Obviamente se va utilizar los mismos nombres en los dos archivos, pues en el archivo “wilo.voca” estamos definiendo estas variables con sus valores y el archivo “wilo.grammar” se crea una estructura usando las variables especificadas anteriormente.

Gramática Julius

Archivo “wilo.grammar”

S: NS_B SENT NS_E
SENT: FUNCION

Figura 39: Archivo .grammar

- S: → Nos indica la estructura del reconocimiento, es aquí donde vamos a establecer la estructura con la que va realizarse el reconocimiento como se indica en la Figura 39.
- SENT: → Nos indica la variable de entrada, pues esta se la utiliza en la estructura de reconocimiento, es decir, SENT es igual a FUNCION.

No olvidemos que las variables NS_B, NS_E y FUNCION ya se definieron en el archivo anterior “wilo.voca”.

Una vez creados estos dos archivos, wilo.grammar y wilo.voca, vamos a utilizar una herramienta de Julius llamada “mkdfa.pl”¹.

¹ herramienta que nos ayuda a realizar conversiones o adaptaciones de archivos de HTK a un formato reconocido por Julius, como el diccionario, gramática

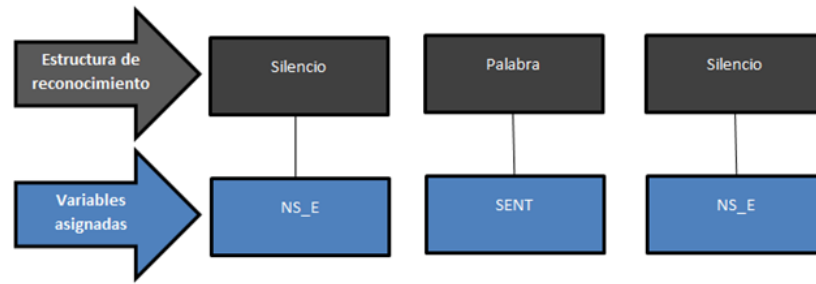


Figura 40: Estructura de reconocimiento Julius

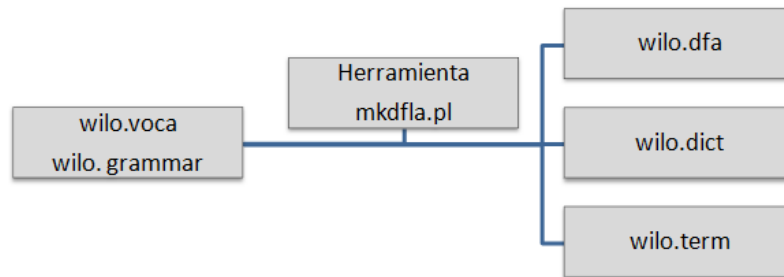


Figura 41: Proceso de la herramienta mkdflla.pl

Para su uso correcto nos ubicamos en el directorio donde se encuentran los archivos (se recomienda que sea un directorio únicamente con los dos archivos) y se ejecuta la siguiente línea:

```
mkdflla.pl wilo
```

Dicho comando realiza una compilación de estos dos archivos por lo que se debía colocar el mismo nombre para los dos archivos. El resultado de la herramienta utilizada es la creación de 3 nuevos archivos: *wilo.dfa*, *wilo.dict* y *wilo.term*, como se indica en la Figura 41, de los cuales vamos a utilizar dos de ellos *wilo.dfa* y *wilo.dict*, colocándolos como parámetros en el archivo de configuración respectivamente, como se indica en la Figura 35.

Como se detalla a continuación:

- *-dfa* → seguido de “-dfa” colocamos un espacio y este seguido del nombre del archivo dfa creado, en nuestro caso es: “-dfa *wilo.dfa*” es así como se estableció en el archivo de configuración de Julius “*configuración.jconf*”.
- *-v* → este parámetro indica que vamos a definir el nombre de nuestro archivo “.dict” en nuestro caso sería “-v *wilo.dict*”.

Creación del HMM general para Julius

Este es el paso que contiene lo más importante para el proceso de reconocimiento mediante Julius, pues debemos establecer un archivo que contenga los modelos entrenados en el proceso HTK.

Aquí interviene el parámetro “-h” en el archivo de configuración de Julius, el cual indica el nombre del archivo que contiene todos los valores de todos los modelos entrenados HTK.

En el siguiente paso se describirá como crear este archivo que necesitamos para definirlo en el parámetro “-h” seguido del nombre del archivo a crear.

Proceso de creación del archivo que contiene las definiciones de los HMM entrenados.

Creamos un archivo nuevo con cualquier editor de texto.

- Creamos una cabecera para el mismo, la cual se la debe establecer tomando la cabecera de cualquiera de los modelos entrenados, y pegarla en nuestro nuevo archivo [33]:

```
~o
<STREAMINFO> 1 39
<VECSIZE> 39
<NULLD><MFCC_D_A_0><DIAGC>
```

A continuación podemos observar un ejemplo del MOM entrenado con HTK, de una de las palabras (abajo)

```
Cabecera {
  ~o
  <STREAMINFO> 1 39
  <VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC>
}

Cuerpo {
  ~h "arriba"
  <BEGINHMM>
  <NUMSTATES> 6
  <STATE> 2
  <MEAN> 39
  -6.054260e+00 -7.084527e+00 -7.585009e+00 -5.030091e+00 -3.865933e+00 -
  1.105901e+00 -3.130810e+00 -1.065020e+00 -1.935943e+00 -2.357102e+00 -
  6.254047e+00 -6.447698e+00 7.285849e+01 1.393288e-01 1.238935e-01 3.381725e-01
  1.702205e-01 -5.288432e-01 -4.706950e-01 -2.637359e-01 -1.568447e-01 -2.078407e-01
  -8.955256e-02 8.863626e-02 -7.963281e-02 4.328485e-01 -6.884302e-02 5.528354e-02
  1.369054e-01 5.133717e-02 -9.948620e-02 -9.029125e-02 7.333789e-03 4.121284e-02 -
  6.087470e-02 -2.000473e-02 1.089843e-01 5.206957e-02 -1.141088e-01
  <VARIANCE> 39
  2.462582e+01 5.224424e+01 2.716679e+01 4.095051e+01 5.069090e+01 3.845806e+01
  3.588802e+01 3.572694e+01 4.888646e+01 8.717096e+01 3.724266e+01 3.719810e+01
  7.002107e+01 1.498954e+00 2.226362e+00 1.663335e+00 2.510794e+00 2.510425e+00
  2.380356e+00 2.452104e+00 3.083138e+00 2.347832e+00 3.141951e+00 2.213857e+00
  2.129194e+00 3.627338e+00 2.354192e-01 3.392318e-01 2.547165e-01 3.757700e-01
  4.154108e-01 4.163989e-01 4.238613e-01 5.215210e-01 4.420945e-01 4.904771e-01
  3.711748e-01 3.254497e-01 3.842576e-01
  <GCONST> 1.189688e+02
  <STATE> 3
  ...
  <STATE> 5
  ...
  <TRANSP> 6
  0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
  0.000000e+00 9.591143e-01 4.088580e-02 0.000000e+00 0.000000e+00 0.000000e+00
  0.000000e+00 0.000000e+00 9.115993e-01 8.630458e-02 2.096134e-03 0.000000e+00
  0.000000e+00 0.000000e+00 0.000000e+00 9.227875e-01 7.598691e-02 1.225632e-03
  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 9.373823e-01 6.261765e-02
  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
  <ENDHMM>
}
```

Figura 42: MOM entrenado con HTK

Luego de esta cabecera debemos extraer todos los cuerpos de los MOM entrenados, en la Figura 42 podemos observar la parte que hemos denominado cuerpo del MOM.

Por lo tanto, en primer lugar colocamos una sola cabecera al inicio del archivo y posteriormente todos los cuerpos de los modelos extraídos, seguido uno del otro sin sus cabeceras.

Cabe recalcar que este archivo contendrá los cuerpos de todos los archivos incluyendo el silencio “sil”.

4.4.9 *Palabras no incluidas en el Diccionario*

Las palabras entrenadas siempre van a ser una cantidad N dentro de cualquier sistema de reconocimiento de palabras aisladas, por lo que existe la posibilidad de que se pronuncien palabras no entrenadas y de este modo puede presentarse una confusión o error en el reconocimiento de voz, es decir, tratamos con palabras fuera de nuestro diccionario.

Modelos Basura

Dentro del proceso de reconocimiento existe un inconveniente que quizá no se lo tome en cuenta en el inicio del entrenamiento HTK: cuando se realizan las pruebas de reconocimiento mediante el micrófono con alguna palabra no entrenada, puede existir confusión por parte del sistema y reconocer la palabra que mas se asemeje a la pronunciada, es decir, existirá un reconocimiento incorrecto.

Es importante mencionar que si no se ha pronunciado palabra del diccionario no debería dar como respuesta alguna de las palabras entrenadas.

Para este caso hemos encontrado de cierto modo una solución, se trata de realizar el entrenamiento del mayor número de palabras posibles, es decir, crear modelos que nos reduzcan el porcentaje de error que se puede presentar y se los ha denominado “modelos basura”.

Estos modelos nos ayudarán a que si se pronuncia alguna palabra de las que no nos interesa para nuestro prototipo, va a reconocer alguna de nuestros “modelos basura”, resultando que no va a afectar a la funcionalidad de nuestro prototipo.

Obviamente, mientras mayor sea el número de “modelos basura” creados, menor será la probabilidad de error o confusión cuando se pronuncien palabras no funcionales para nuestro prototipo.

Proceso Modelos Basura

Estos modelos no son más que los modelos MOM como se han explicado hasta el momento, es decir, se los debe procesar de la misma manera, con su creación, inicialización, y re-estimación.

Como podemos observar en la Figura 42, el proceso de un “Modelo Basura” es similar a un modelo funcional del prototipo, pero con la diferencia de que en el momento que el sistema reconozca alguna palabra de los modelos basura, no va a realizar ninguna acción nuestro sistema, es decir, solo va a existir el reconocimiento pero no se activará ninguna función en el sistema operativo.

De este modo, si se pronuncia alguna palabra desconocida y esta se encuentra como modelo basura, la reconocerá y no sucederá ninguna acción, reduciendo la posibilidad de que la palabra desconocida sea confundida con una palabra funcional del prototipo.

Ejecución Julius

Una vez que tenemos todos los parámetros necesarios para un correcto funcionamiento mediante Julius, su llamada mediante el terminal es

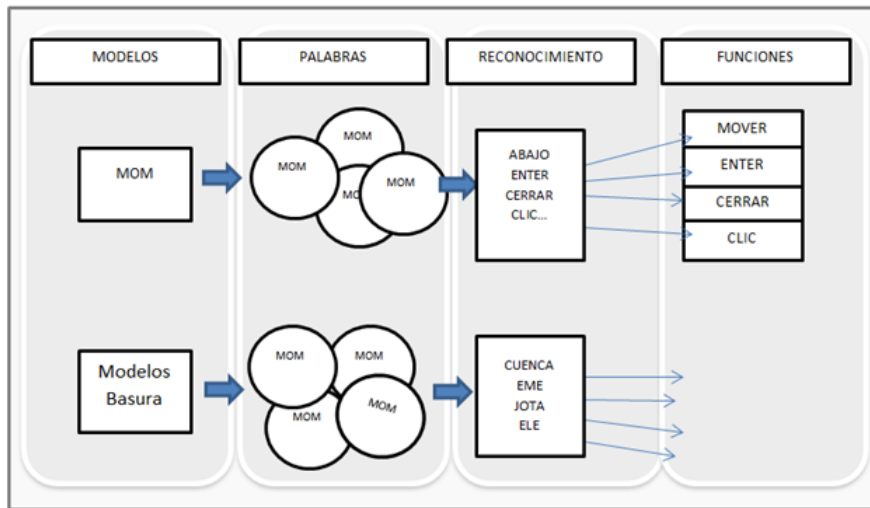


Figura 43: Función Modelos Basura

sencilla [33] [10]:

Julius -C configuracion.jconf

Donde:

- -C → Indica el nombre de nuestro archivo de configuración "jconf", en nuestro caso "-C configuración.jconf".

Si todo está correcto, Julius empezará a escuchar en tiempo real, y apenas detecte algún ruido o señal de voz, lanzará o imprimirá algún resultado, todo esto dependiendo de los parámetros de entrada que hemos realizado en los procesos anteriores.

4.4.10 Acoplamiento del Sistema de Reconocimiento al Prototipo Mediante Java

Para un acoplamiento correcto que permita cumplir con las funcionalidades planteadas en nuestro prototipo, hemos utilizado clases Java que proveen métodos necesarios para llamadas y control de funciones de nuestro sistema operativo.

Clase Runtime Java [6]

Esta clase nos sirve para iniciar un ejecutable externo desde Java, de este modo podemos llamar a nuestro sistema de reconocimiento previamente entrenado (HTK+JULIUS).

El código básico para hacer una llamada es el siguiente:

En nuestro caso debemos hacer la llamada al proceso de Julius que ya hemos definido anteriormente, es decir:

Julius -C configuracion.jconf

Quedando de la siguiente manera:

Algoritmo 4.1

```

try {
    Process p = Runtime.getRuntime().exec ("firefox.exe");
}
catch (Exception e){

    /* Excepción en caso de error. */
}

```

Luego obtenemos la salida del comando mediante el método **getInputStream** de la clase **InputStream**.

Algoritmo 4.2

```

try {

    Process p = Runtime.getRuntime().exec ("Julius -C
        configuracion.jconf ");
}

catch (Exception e) {
    /* Excepción en caso de error. */
}

```

El siguiente paso es utilizar el método **readLine()** de la clase **BufferedReader** de Java, este método devuelve automáticamente un **String** con lo resultados de salida del ejecutable.

En nuestro caso devolverá las palabras reconocidas. Una vez obtenido el **String** podemos realizar las comparaciones necesarias y realizar las funciones que se requieran.

GetInputStream() → obtiene la salida del programa.

BufferedReader → sirve para poder leer la salida usando un gestor basado en ficheros de texto secuenciales.

Ahora simplemente para leer la primera línea lo podemos realizar mediante la siguiente instrucción:

```
String variable = br.readLine();
```

Funcionalidad de las Palabras Reconocidas

Una vez obtenidos los datos de reconocimiento, el siguiente paso sería generar las condiciones que nos ayuden al control de ventanas y aplicaciones. Con ello podremos mover el puntero del mouse con comandos de voz de acuerdo al entrenamiento previo. De igual manera, podremos establecer funciones del teclado de acuerdo a nuestras órdenes de voz, pudiendo así lograr uno de los objetivos de nuestro prototipo, controlar el computador con nuestra voz. Para realizar la siguiente tarea necesitamos emplear la clase **Robot** de Java.

Clase Robot Java [9]

Es una clase que se encuentra en el paquete **AWT**, que sirve para automatizar la invocación de las funciones del teclado y mouse. De

este modo podemos emular cualquier actividad del usuario. En nuestro caso vamos a utilizar lo que se refiere al movimiento del puntero del mouse, el evento de clic, y funcionalidades de ciertas teclas. Por ejemplo, emplearemos comandos para poder manejar un menú, acceder a botones, abrir y cerrar ventanas, activar y desactivar funciones y órdenes básicas que se puede realizar con las mismas operaciones.

Movimiento del puntero del Mouse [12]

El mouse es de suma importancia para manejar cualquier sistema operativo dentro de la interfaz gráfica del mismo. Entonces, si logramos hacer esto sería de mucha ayuda en el control de cualquier interfaz que se nos presente.

A continuación vamos a indicar la forma básica para realizar lo mencionado.

```
Robot robot = new Robot();
robot.mouseMove(int x, int y);
```

Este es el código utilizado para mover el puntero del mouse, a una posición (x,y), como podemos observar es un código sencillo, y se debe partir de la creación de una instancia de la clase Robot. Luego hacemos la llamada al método de la clase Robot, llamado "mouseMove" el cual recibe como parámetro dos valores enteros que representan la ubicación dentro del plano (x,y) de la pantalla, pudiendo moverlo a cualquier ubicación según sea necesario.

Simulación Eventos del Mouse [12]

Una utilidad de mucha importancia para quien desea trabajar mediante el mouse, es la que permite emplear las funciones de sus botones, es decir, clic, doble clic, botón central o scroll:

Clic izquierdo

```
robot.mousePress(InputEvent.BUTTON1_MASK);
robot.mouseRelease(InputEvent.BUTTON1_MASK);
```

El método "mousePress" de la clase Robot sirve para simular un clic y el método "mouseRelease" sirve para simular que se suelta el botón del mouse.

Clic derecho

Para simular el evento del clic derecho simplemente se debe cambiar el parámetro de los métodos de la clase Robot, como podemos observar a continuación:

```
robot.mousePress(InputEvent.BUTTON2_MASK);
robot.mouseRelease(InputEvent.BUTTON2_MASK);
```

Simulación Teclado [12]

De igual manera, podemos realizar una simulación de las funciones de cada una de las teclas existentes.

Algoritmo 4.3

```

try {
    Robot robot = new Robot();
    robot.keyPress(KeyEvent.VK_H);
    robot.keyPress(KeyEvent.VK_O);
    robot.keyPress(KeyEvent.VK_L);
    robot.keyPress(KeyEvent.VK_A);
}
catch (AWTException e) {
    e.printStackTrace();
}

```

Con el metodo `keyPress(KeyEvent.A)` donde A puede ser cualquier valor de acuerdo a la asignación de cada una de las teclas. Por ejemplo, en el caso anterior se está escribiendo la palabra HOLA. En nuestro caso podemos utilizar varias combinaciones de teclas para nuestras funciones, para poder cerrar cualquier tipo de ventanas, aplicaciones, mediante las siguientes líneas:

```

robot.keyPress(KeyEvent.VK_ALT);
robot.keyPress(KeyEvent.VK_F4);

```

De igual manera:

```

robot.keyPress(KeyEvent.VK_ENTER);

```

Nos puede ser de mucha utilidad la tecla “ENTER” para un sinnúmero de funciones en varias aplicaciones o acceso a ellas, pues mediante tecla podemos tener acceso a algún botón u opción seleccionada.

De este modo, tenemos las herramientas necesarias para acoplar nuestro sistema de reconocimiento del habla, ayudándonos con clases brindadas por Java para realizar un prototipo mucho más fácil de utilizar y controlar para el usuario.

Sistema Final

Luego de terminar con los procesos de cada una de las herramientas, efectuando a cabalidad lo detallado, tendremos nuestro prototipo listo, que cumpla con los objetivos funcionales establecidos inicialmente.

Nuestro sistema será capaz de recibir órdenes mediante comandos de voz y realizar las funciones que se han asignado a cada comando, permitiéndonos tener un control de nuestro sistema operativo sin ningún tipo de contacto con el hardware de nuestro ordenador.

Por ejemplo, mediante nuestro prototipo podremos mover el puntero del mouse mediante comandos de voz, al igual que cerrar ventanas, dar clic, Enter, activar y desactivar funciones, simular teclas de dirección, tecla de tabulación, simular escritura de ciertas palabras, terminar el proceso del prototipo. Comandos que de cierta manera nos permiten un control básico del sistema operativo, mediante los cuales podemos realizar otras actividades dependiendo del software existente.

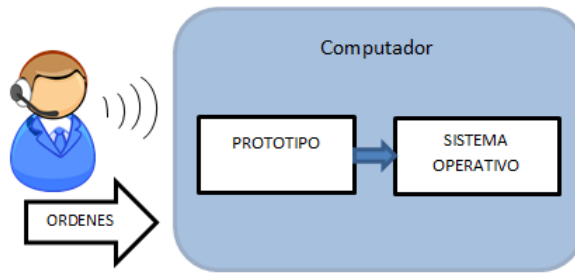


Figura 44: Prototipo Final

Funcionamiento del prototipo

Posee una interfaz gráfica sencilla, la cual sirve principalmente para iniciar y terminar el proceso de reconocimiento, como podemos observar en la siguiente Figura:



Figura 45: Interfaz Gráfica

El proceso para manejar la aplicación es sencillo, pues mediante los botones definidos podemos llamar y ejecutar nuestros eventos, a través de los cuales se encuentran los procesos de reconocimiento elaborados anteriormente.

El botón “Inicio” nos da paso a un proceso, presentándonos un sistema de reconocimiento de voz, con el cual podemos controlar nuestro sistema operativo, de las siguientes órdenes.

Movimiento del puntero del mouse.

Hemos creído factible poder realizar esta operación, pues el mouse es de mucha importancia para el control de cualquier ordenador.

En este punto hemos definido las siguientes palabras:

PALABRA	FUNCIÓN
Derecha	Movimiento del puntero del mouse hacia la derecha de la pantalla
Izquierda	Movimiento del puntero del mouse hacia la izquierda de la pantalla
Arriba	Movimiento del puntero del mouse hacia la parte superior de la pantalla
Abajo	Movimiento del puntero del mouse hacia la parte inferior de la pantalla
Clic	Simula un clic izquierdo del mouse

Cuadro 2: Palabras que simulan las funciones del Movimiento del Puntero

Activa funciones

También existe la posibilidad de activar funciones, por ejemplo con la palabra "Teclado", se activarán las mismas palabras anteriores de dirección pero esta vez con diferentes funciones .

PALABRA	FUNCIÓN
Teclado	Activa funciones del teclado (botones de dirección)
Derecha	Simula tecla de dirección "derecha"
Izquierda	Simula tecla de dirección "izquierda"
Arriba	Simula tecla de dirección "arriba"
Abajo	Simula tecla de dirección "abajo"
Escritura	Activa la función de escritura

Cuadro 3: Palabras que permiten Activar la Función de las teclas de Dirección

Y si se vuelve a mencionar la palabra "teclado", se desactivarán estas funciones y volverán a las funciones de movimiento del puntero del mouse.

Al momento de pronunciar la palabra Escritura se activa la función de escritura, es decir, al momento que diga cualquier palabra de las entrenadas, y nos encontramos ubicados en cualquier editor de texto, el sistema va a escribir como texto la palabra mencionada.

Teclado

También se han establecido órdenes que simulen ciertas teclas, con un previo análisis hemos considerado que las siguientes son importantes para un control básico.

PALABRA	FUNCIÓN
Enter	Simula la tecla "Enter", con la cual podemos tener acceso a diferentes funciones dentro del sistema operativo.
Siguiente	Simula la tecla "Tabulador", con la cual podemos cambiar de opciones de distintas aplicaciones.
Salir	Simula la combinación de teclas "Alt + F4", con la cual podemos salir de diferentes ventanas, o aplicaciones.

Cuadro 4: Palabras que activan la función de ciertas teclas para el control del S.O.

Otras funciones

De igual manera, se han establecido otras palabras que representen nuevas funciones que nos pueden ser de ayuda al momento del control del sistema operativo.

PALABRA	FUNCIÓN
Gracias	Termina la ejecución del prototipo

Cuadro 5: Palabra que permite terminar el proceso de Ejecución

Como podemos observar, se han detallado las órdenes que se le puede dar a nuestro prototipo, mediante las cuales hemos realizado pruebas teniendo un nivel aceptable de control, (este aspecto se analiza a detalle en el siguiente punto).

Parte V

RESULTADOS Y DISCUSIÓN

RESULTADOS Y DISCUSIÓN

En este apartado describiremos la etapa final, detallando el esquema de funcionamiento de nuestro prototipo completo, mediante el cual se ha alcanzado los objetivos planteados inicialmente.

Asimismo hemos logrado realizar el prototipo con la mejor eficacia posible, basándonos en la experiencia de práctica y pruebas.

También describiremos los resultados obtenidos, mediante los cuales se hará una fácil observación de los niveles de respuesta en cuanto al reconocimiento de nuestro sistema, esta parte será siempre importante para medir lo que el sistema es capaz de realizar, mediante los procesos del proyecto.

Finalmente, tomando como base lo anterior, se plantea el modelo de mejoras que podríamos realizar al sistema, considerando que el objetivo principal es explorar todas las potencialidades del reconocimiento automático del habla a través de HMM.

5.1 EJECUCIÓN DEL PLAN DE EXPERIMENTACIÓN

El plan de experimentación principalmente se lo realiza mediante el módulo de pruebas de reconocimiento de voz, que consta básicamente de la utilización de Julius, que maneja los modelos entrenados con HTK de cada una de las palabras.

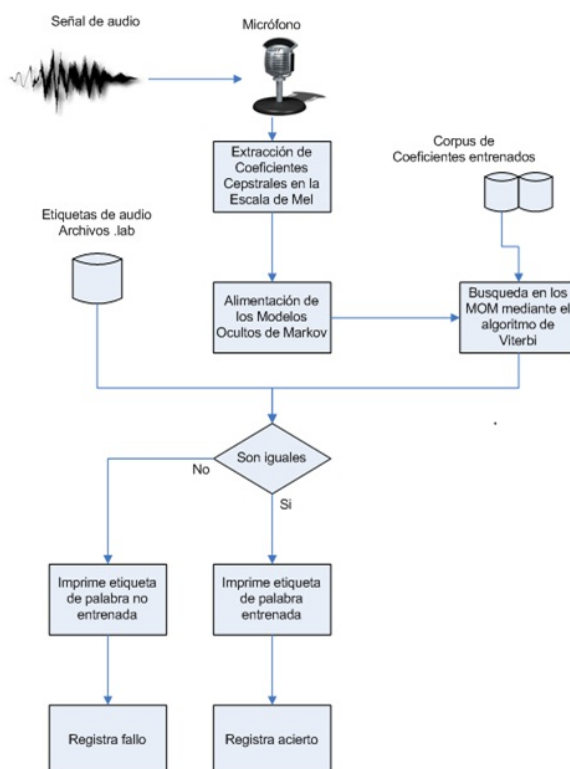


Figura 46: Proceso de experimentación del reconocimiento de palabras

Ejecución del modulo de pruebas mediante Julius.

Para proceder a evaluar el rendimiento de los MOM en el prototipo, vamos a fijarnos en el nivel de reconocimiento que brinde el sistema mediante Julius, para que a la vez podamos conseguir reconocimiento del habla en tiempo real.

Obviamente debemos tener mucha atención en los parámetros que se utilizan para la ejecución de Julius, pues de esto también van a depender los resultados.

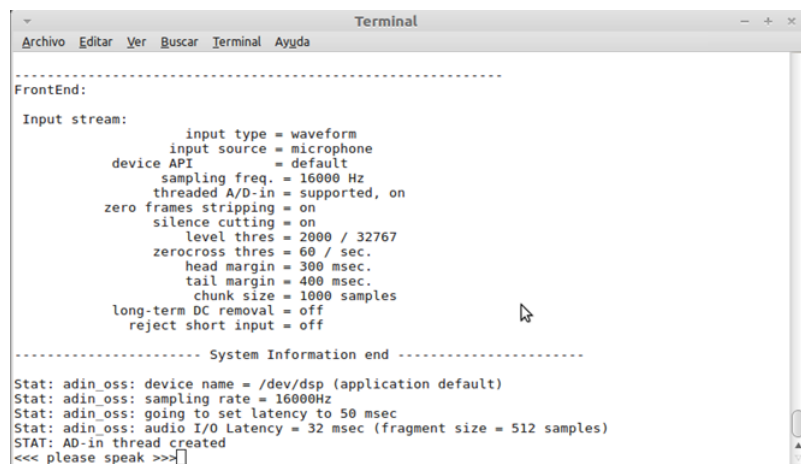
Julius podemos ejecutarlo desde el terminal de nuestro sistema operativo GNU/Linux de la siguiente manera:

Julius -C parametros.jconf

Donde el archivo “parametros.jconf” contiene los parámetros con los que va a ser ejecutado Julius como se ha explicado en diseño del prototipo dentro del capítulo 4.

Cuando ejecutemos la llamada de Julius correctamente, debemos obtener un resultado como se indica en la figura 47, entonces procedemos a realizar las pruebas de cada una de las palabras entrenadas para ello y generará el reconocimiento.

Cuando se ha realizado el reconocimiento correcto de las palabras, el resultado que se obtiene es el que se observa en la figura 48, y de esta manera Julius seguirá escuchando continuamente hasta que se detenga el proceso, permitiéndonos realizar las pruebas con el número de palabras que se desee como se indica en la figura 49.



```

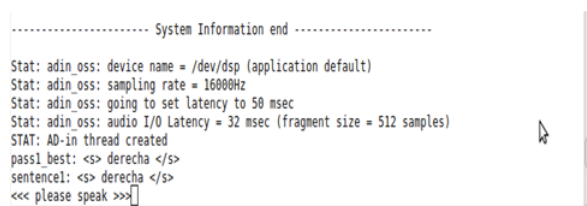
-----
FrontEnd:

Input stream:
    input type = waveform
    input source = microphone
    device API = default
    sampling freq. = 16000 Hz
    threaded A/D-in = supported, on
    zero frames stripping = on
    silence cutting = on
    level thres = 2000 / 32767
    zerocross thres = 60 / sec.
    head margin = 300 msec.
    tail margin = 400 msec.
    chunk size = 1000 samples
    long-term DC removal = off
    reject short input = off

----- System Information end -----
Stat: adin_oss: device name = /dev/dsp (application default)
Stat: adin_oss: sampling rate = 16000Hz
Stat: adin_oss: going to set latency to 50 msec
Stat: adin_oss: audio I/O Latency = 32 msec (fragment size = 512 samples)
STAT: AD-in thread created
<<< please speak >>>

```

Figura 47: Ejecución de la herramienta Julius

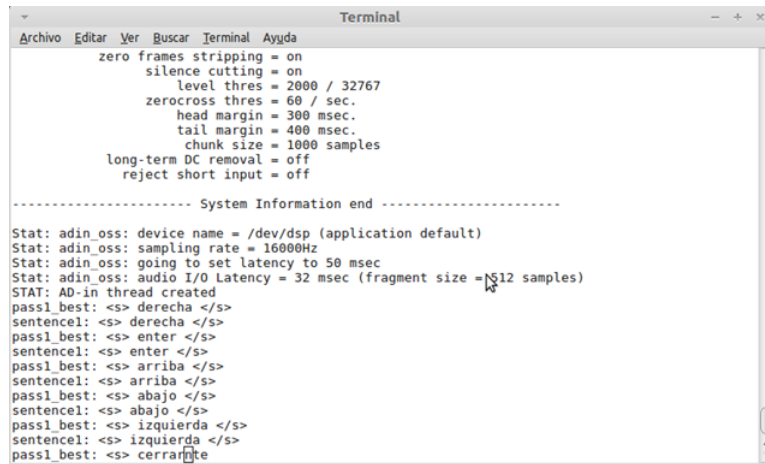


```

----- System Information end -----
Stat: adin_oss: device name = /dev/dsp (application default)
Stat: adin_oss: sampling rate = 16000Hz
Stat: adin_oss: going to set latency to 50 msec
Stat: adin_oss: audio I/O Latency = 32 msec (fragment size = 512 samples)
STAT: AD-in thread created
pass1 best: <s> derecha </s>
sentencel: <s> derecha </s>
<<< please speak >>>

```

Figura 48: Reconocimiento de una palabra mediante Julius



```

Terminal
Archivo Editar Ver Buscar Terminal Ayuda
zero frames stripping = on
silence cutting = on
level thres = 2000 / 32767
zerocross thres = 60 / sec.
head margin = 300 msec.
tail margin = 400 msec.
chunk size = 1000 samples
long-term DC removal = off
reject short input = off

----- System Information end -----

Stat: adin_oss: device name = /dev/dsp (application default)
Stat: adin_oss: sampling rate = 16000Hz
Stat: adin_oss: going to set latency to 50 msec
Stat: adin_oss: audio I/O Latency = 32 msec (fragment size = 512 samples)
STAT: AD-in thread created
pass1_best: <s> derecha </s>
sentencel: <s> derecha </s>
pass1_best: <s> enter </s>
sentencel: <s> enter </s>
pass1_best: <s> arriba </s>
sentencel: <s> arriba </s>
pass1_best: <s> abajo </s>
sentencel: <s> abajo </s>
pass1_best: <s> izquierda </s>
sentencel: <s> izquierda </s>
pass1_best: <s> cerrar </s>

```

Figura 49: Pruebas continuas mediante Julius

Se realizará la ejecución del plan de experimentación de nuestro prototipo con pruebas en vivo con la ayuda de 10 personas, tanto hombres como mujeres.

Cada persona pronunciará una lista de 15 palabras entrenadas y 10 palabras no entrenadas, es decir, en total se realizarán 250 pruebas de reconocimiento de voz, comprobando y midiendo así la efectividad de reconocimiento, en un ambiente adecuado que permita realizar las pruebas de una forma real. Cabe recalcar que para todas las pruebas se ha utilizado el mismo hardware, es decir, el mismo computador y micrófono, para obtener mejores resultados.

Se les ha pedido a las personas que pronuncien una lista de las palabras entrenadas y no entrenadas, escogiendo las palabras al azar, para comprobar el nivel de reconocimiento y consecuentemente de error o confusión que pueda existir en el sistema, ya que es importante observar la reacción del prototipo ante cualquier señal que pueda presentarse.

El siguiente grupo de palabras fue presentado a las personas para que las pronuncien en el mismo orden: Arriba, Abajo, Enter, Arriba, Derecha, Cerrar, Siguiente, Abajo, Gracias, Izquierda, Izquierda, Clic, Cerrar, Clic, Teclado. Y la lista de las palabras no entrenadas: Dólar, mico, orate, televisión, radio, botella, universidad, pato, proyecto, esfero.

Se les ha indicado a las personas que efectuaron las pruebas, que deben realizar la pronunciación de una forma correcta, es decir, emplear un correcto tono de voz, mantenerse a una distancia adecuada del micrófono y pronunciar palabra por palabra con una pequeña pausa para observar el resultado luego de cada prueba sin ningún inconveniente.

De este modo evaluaremos la efectividad del sistema de reconocimiento, y consecuentemente de los modelos basura que se han planteado dentro del prototipo, es decir, en que porcentaje nos ayudan dichos modelos en cuanto a la reducción de la probabilidad de confusión en el reconocimiento, cuando se pronuncien palabras no entrenadas.

Los resultados de estas pruebas se detallan en el siguiente apartado.

5.2 ANÁLISIS DE RESULTADOS

En este punto detallaremos los resultados que se obtuvieron en la ejecución del plan de experimentación.

Hemos tomado los resultados directamente de la consola de GNU\Linux comprobando los aciertos o errores que se presenten en cada una de las pruebas.

Esto consiste en medir el porcentaje de precisión de reconocimiento de las palabras pronunciadas.

De las 10 personas con las que se realizaron las pruebas, 5 son hombres y 5 mujeres, para observar un resultado más real y certero.

Cabe mencionar que todas las personas son mayores de edad, puesto que es muy importante saber que se trata de tonalidades de voz de personas adultas, al igual que en el corpus de entrenamiento inicial.

Precisión

A continuación se detalla una tabla con los valores de cada una de las pruebas que han sido realizadas.

	Sexo	Palabras Pronunciadas	Palabras reconocidas	Palabras Fallidas	Porcentaje de aciertos	Porcentaje de error
1	Masculino	15	15	0	100	0
2	Masculino	15	14	1	93,33333333	6,66666667
3	Masculino	15	13	2	86,66666667	13,33333333
4	Masculino	15	13	2	86,66666667	13,33333333
5	Masculino	15	13	2	86,66666667	13,33333333
6	Femenino	15	13	2	86,66666667	13,33333333
7	Femenino	15	12	3	80	20
8	Femenino	15	14	1	93,33333333	6,66666667
9	Femenino	15	12	3	80	20
10	Femenino	15	13	2	86,66666667	13,33333333
				TOTAL:	88	12

Cuadro 6: Precisión de reconocimiento de las palabras

Observamos fácilmente que existe un mejor nivel de efectividad en lo que se refiere a voces masculinas, esto se debe a que nuestra base de datos de entrenamiento posee mayor cantidad de muestras de voz masculinas.

Obviamente, lo ideal sería definir un corpus de entrenamiento con la mayor cantidad de voces de diferentes personas, tanto masculinas como femeninas, para establecer una igualdad de reconocimiento.

	Hombre	Mujer
Aciertos	90,66666667	85,33333333
Errores	9,333333333	14,66666667

Cuadro 7: Nivel de Reconocimiento entre hombres y mujeres

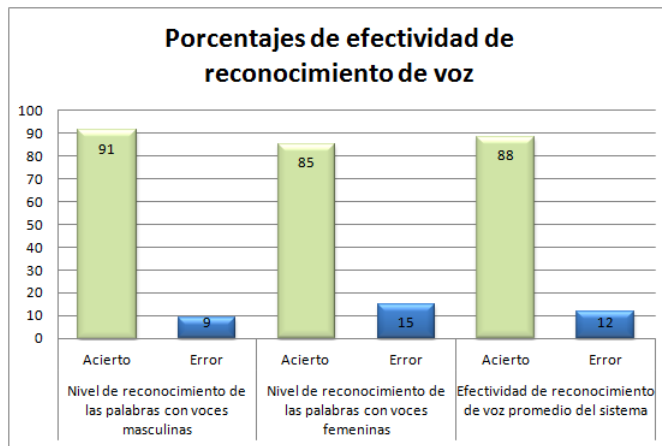


Figura 50: Nivel de reconocimiento de voz obtenido en las pruebas realizadas

Podemos observar que existe un porcentaje de aciertos muy bueno, tomando en cuenta que al momento de las pruebas no existía un ambiente completamente adecuado es decir, no existía un silencio absoluto, u otros aspectos negativos, como una pronunciación incorrecta de las palabras, aspectos que disminuyeron los aciertos.

Mediante las pruebas realizadas hemos podido observar un 88 % de precisión, y consecuentemente un 12 % de error en nuestro prototipo, tomando en cuenta los siguientes aspectos negativos que se pueden presentar por lo cual existe un porcentaje considerable de error:

- Dentro de todas las muestras de voz de nuestro corpus de entrenamiento, no existe una equidad, ya que la mayoría de voces pertenecen a hombres, por lo que se presentará una efectividad mucho mejor cuando se realice las pruebas por una persona de sexo masculino, debido a la similitud de tonalidad de voz.
- Depende del ambiente en que se realice las pruebas.
- La claridad de las palabras mencionadas, es decir, una pronunciación correcta, fuerte y clara.
- Micrófono con el que se está realizando las pruebas, es decir, si el micrófono tiene reducción de ruido, claridad y un buen rendimiento.

Pues sabemos que HTK es una herramienta muy poderosa en cuanto al reconocimiento, entonces puede depender de factores ajenos para su rendimiento.

Precisión de palabras no entrenadas

Como hemos tratado en el capítulo 4 en el apartado de “Palabras no incluidas en el diccionario”, dentro de los pasos de desarrollo de nuestro prototipo hemos incluido los modelos basura, para ayudar al rendimiento, puesto que existe la posibilidad de que se pronuncien palabras no entrenadas en el proceso de reconocimiento, por lo que el sistema puede presentar una confusión y realizar alguna función u orden no dada.

Para ello hemos realizado pruebas con palabras no entrenadas, a fin de verificar la respuesta del sistema frente a esta circunstancia.

Palabra No entrenada	Confusión/ Error	Palabra reconocida
dólar	no	hola
mico	no	quito
orate	no	hola
televisión	si	inicio
radio	no	Vladimir
botella	si	izquierda
universidad	si	inicio
pato	no	gato
proyecto	no	texto
esfero	no	perro

Cuadro 8: Ejemplo de pruebas con palabras no entrenadas

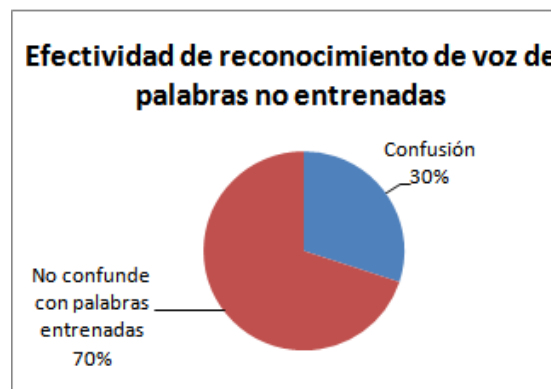


Figura 51: Nivel de Reconocimiento de Palabras No entrenadas

La técnica del uso de los modelos basura nos ayuda a reducir la probabilidad de confusión de nuestro prototipo cuando se trata de palabras no entrenadas, por lo que de acuerdo a la cantidad y las palabras asignadas de los modelos basura va a depender el porcentaje de confusión que se obtenga.

En el cuadro 9 se puede apreciar con claridad un ejemplo de las pruebas realizadas a una persona con las 10 palabras no entrenadas, dándonos cuenta que nos ha sido de ayuda al reducir un cierto porcentaje de confusión en el momento de realizar pruebas con palabras no entrenadas, pues se puede decir que existe confusión por parte del prototipo al momento que se reconozca alguna palabra que realice alguna función dentro del prototipo.

Por ejemplo, se reconoce la palabra “Izquierda” si se ha pronunciado la palabra no entrenada “botella”, entonces se puede decir que existió error o confusión, pues sabemos que la palabra “izquierda” si realiza una función dentro de nuestro prototipo, pero si se pronuncia la palabra

“botella” y reconoce alguna palabra basura como “hola” entonces se puede decir que el método de los modelos basura nos ha servido pues no está afectando el rendimiento funcional de nuestro prototipo, ya que la palabra “hola” no cumple ninguna función dentro del programa.

Entonces, si no existiera ningún modelo basura siempre existiría confusión, ya que en el momento del reconocimiento el prototipo siempre va a buscar la palabra que se asemeje más a la pronunciada y nos tiene que devolver un resultado.

Sujeto de Experimentación	Palabras No entrenadas pronunciadas	Palabra Confundidas
1	10	3
2	10	3
3	10	4
4	10	5
5	10	5
6	10	4
7	10	4
8	10	5
9	10	4
10	10	5

Cuadro 9: Resultado de pruebas con palabras No encontradas

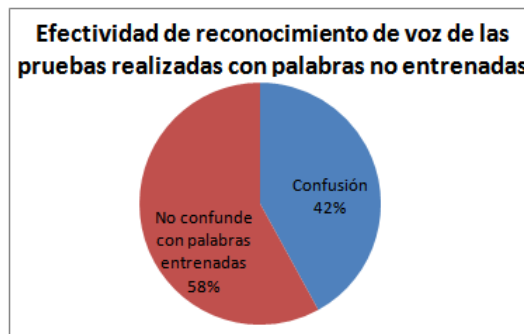


Figura 52: Resultado de pruebas con palabras no entrenadas

En la Figura 52 podemos observar el porcentaje de reducción de error o confusión que se tiene gracias a los modelos basura, obviamente un porcentaje existente dentro de ese grupo de palabras no entrenadas que han sido mencionadas para las pruebas, puesto que este valor como porcentaje va a depender de las palabras desconocidas que se pronuncien.

5.3 ESPECIFICACIÓN DE MEJORAS PLANTEADAS A LA EXPERIMENTACIÓN

La principal característica que se puede mejorar en el sistema y consecuentemente mejorará la efectividad, es el corpus de entrenamiento inicial.

Pues el entrenamiento realizado mediante HTK va a ser mucho mejor mientras se lo realice con una cantidad mayor de muestras de voz, obviamente también va a depender de la calidad de las mismas, es decir, pronunciación correcta, ambiente de grabación adecuado, etc.

Para realizar un sistema mucho más robusto, podemos trabajar de igual manera con muestras tomadas de voces con tonalidades diferentes, para de este modo abarcar un reconocimiento mucho más amplio.

Estos dos aspectos deben ser tomados en cuenta al inicio del proceso, ya que la base de datos de entrenamiento que debemos obtener es primordial y el paso principal para la calidad de reconocimiento.

Dentro de los puntos anteriores también nos podemos referir a un punto importante, que es el de los “Modelos Basura”, pues al igual que en los modelos principales, mientras mayor sea el número de “modelos basura” incluidos en nuestro prototipo, menor será la probabilidad de confusión de nuestro sistema, cuando se pronuncie una palabra no entrenada. Otra de las mejoras que se le puede hacer al prototipo desarrollado, es la utilización de modelos híbridos basados en la unión de los Modelos Ocultos de Markov con Redes Neuronales.

Esta unión permitirá realizar un mejor reconocimiento de la voz debido a que el prototipo actual cuenta con la probabilidad de transición entre estados y su probabilidad de emisión, esta última sería de mucha ayuda para poder crear un modelo de Red Neuronal.

La creación de esta red permitirá el análisis por pesos de los fonemas que fueron entrenados y etiquetados en el capítulo 4 y de esta manera poder mejorar los resultados obtenidos.

Como otro punto para mejorar los resultados obtenidos se podría crear una página en la cual las personas voluntariamente dejen grabaciones de sus voces, y tomarlas para nuestra base de datos o corpus de entrenamiento inicial y de este modo sería de mucha ayuda para que el prototipo pueda reconocer diferentes tipos de voces, tonalidad, dialecto, etc.

Finalmente, como otra alternativa para mejorar los resultados de este prototipo sería utilizar una nueva herramienta llamada ATK, que es un API diseñado para facilitar la creación de aplicaciones experimentales para HTK. Se compone de una capa de C++ que permite realizar aplicaciones basadas en voz.

Esta herramienta incorpora una funcionalidad de cálculo de medidas de confianza, de manera que si una determinada hipótesis no supera un valor umbral previamente establecido, es posible desechar la hipótesis, permitiéndonos obtener las muestras correctas para poder tener un reconocimiento de los patrones de la voz de la mejor manera.

Se podría tomar en cuenta también para mejorar los resultados el uso de un micrófono con características especiales que nos ayuden a disminuir el ruido al momento de realizar las pruebas en tiempo, las características podrían ser las siguientes:

- Micrófono con cancelación de ruido.
- Cascos cerrados para reducir las interferencias de fondo.
- Micrófono de gran solidez con capacidad de comprensión de voz a gran velocidad.
- Reconocimiento de voz con control de volumen de entrada de línea e interruptor de silenciamiento del micrófono.

Una recomendación para realizar un sistema completo de reconocimiento de voz, es que se podría trabajar con fonemas, en lugar de palabras (como se ha hecho en nuestro caso).

Al momento de realizar un HMM para que represente a diferentes fonemas vamos a incrementar las palabras de reconocimiento, pues mediante estos vamos a tener la posibilidad de unir uno o mas fonemas creando un número N de palabras mediante sus combinaciones, de este modo también se reducirá la posibilidad de confusión de palabras en el reconocimiento.

Si se desea elaborar un sistema mucho más amplio y completo con un diccionario robusto, se debería tomar en cuenta todos los puntos anteriores y de este modo va a existir una mejora muy considerable para el sistema, ya que nos hemos basado en las pruebas realizadas, por lo que se nos ha hecho posible observar estos detalles.

CONCLUSIONES

La efectividad de reconocimiento del prototipo, va a depender altamente de una serie de características de nuestro corpus de entrenamiento como la calidad, cantidad y otros aspectos ajenos a HTK, como el hardware y ambiente de trabajo en el momento de las grabaciones y pruebas realizadas, por lo tanto debemos tener mucho mas énfasis en la recolección de muestras de voz para la creación de nuestro corpus o base de datos de entrenamiento.

Los Modelos Ocultos de Markov, poseen muchas utilidades, y es considerada como una de las mejores opciones al momento de tratar el reconocimiento automático de voz. El tipo de MOM más eficiente para trabajar con el reconocimiento de voz, es el modelo de Bakis o de izquierda a derecha, ya que en este tipo de modelos las transiciones entre estados solo se dan hacia adelante o permanecen en el mismo lugar, por lo que se lo utiliza para modelar señales que varían con el tiempo como en este caso la voz.

Mediante las pruebas realizadas hemos comprobado la efectividad de reconocimiento obtenido, y se puede decir que el trabajar con MOM es una técnica que nos presenta un nivel de reconocimiento bastante aceptable, en este caso se ha obtenido un porcentaje sobre el 85 % de efectividad y con la posibilidad de incrementar este valor de acuerdo al proceso de desarrollo del sistema.

Refiriéndonos específicamente a la efectividad del prototipo, podemos decir que los resultados se darán de una mejor manera si la persona que realiza las pruebas, antes ha incluido sus muestras de voz en el corpus de entrenamiento para HTK, entonces si queremos obtener un sistema para una sola persona o locutor vamos a necesitar muestras únicamente de esta persona, pero si dentro de nuestros objetivos está tener un reconocimiento de una manera mucho mas global, es decir, de diferentes locutores, deberíamos recoger muestras de voz de diferentes personas, para de este modo tener un sistema más efectivo para distintas personas.

El software de libre distribución nos brinda una efectividad muy buena, ya que a más de HTK y el sistema operativo utilizado, hemos manejado otra herramienta que nos ayudo mucho para cumplir nuestros objetivos, llamada Julius. Se trata de una herramienta que tiene la característica de compatibilidad con los Modelos Ocultos de Markov entrenados con HTK y posee una potencialidad muy buena para realizar un reconocimiento de voz en tiempo real continuo, que es lo requerido en el prototipo.

RECOMENDACIONES

Para hacer un sistema de reconocimiento mucho mas robusto y con un mejor rendimiento, se recomienda trabajar con fonemas, es decir, crear un MOM para cada fonema, y de este modo cada palabra reconocida será un conjunto de fonemas, de igual manera esto nos ayudará a reducir la probabilidad de confusión de palabras, evitando así el uso de los modelos basura.

Si se desea tener un mismo nivel de reconocimiento para todas las palabras se debería tener una misma cantidad y calidad de muestras de voz para cada una de ellas en el corpus de entrenamiento, pues estos factores son muy importantes para el rendimiento del prototipo.

Para mejorar la precisión en cuanto al reconocimiento se recomienda trabajar con una herramienta llamada ATK, que se trata de un API diseñado para facilitar la creación de aplicaciones experimentales para HTK, esta herramienta incorpora una funcionalidad de cálculo de medidas de confianza, por lo que si una determinada hipótesis no supera un valor umbral previamente establecido, es posible desechar la hipótesis, permitiéndonos obtener las muestras correctas para poder tener un reconocimiento exacto de los patrones de la voz de la mejor manera y obtener resultados mucho mas exactos, este proceso puede remplazar el uso de los modelos basura al momento de tratarse de palabras no entrenadas.

De igual manera se recomienda trabajar con un micrófono profesional, esto nos ayudará de gran manera incrementar la efectividad de reconocimiento, al anular el ruido externo y recibir una señal de audio mucho mas nítida, permitiendo así obtener valores más exactos para el reconocimiento.

BIBLIOGRAFÍA

- [1] *Modelo determinístico*. URL http://es.wikipedia.org/wiki/Modelo_deterministico. (Cited on page 14.)
- [2] *Dragon NaturallySpeaking*. URL http://en.wikipedia.org/wiki/Dragon_NaturallySpeaking. (Cited on page 49.)
- [3] *Estocástico*. URL <http://es.wikipedia.org/wiki/Estocastico>. (Cited on page 14.)
- [4] *Modelo oculto de Márkov*. URL http://es.wikipedia.org/wiki/Modelo_oculto_de_Markov. (Cited on pages xvii, 13, and 20.)
- [5] *Codificación Perceptual y Oído Humano*. URL <http://www.reypastor.org/departamentos/dinf/enalam/sonido/mp3.htm>. (Cited on page 71.)
- [6] *Ejemplos java y C/Linux*. URL <http://www.chuidiang.com/java/ejemplos/Runtime/runtime.php>. (Cited on page 91.)
- [7] *Hidden Markov Model Toolbox*. Universidad de Washington. URL http://brl.ee.washington.edu/Research_Active/Research_Active/Surgery/Device_HMM/HMM.html. (Cited on page 45.)
- [8] *Preénfasis*. URL <http://es.wikipedia.org/wiki/Pre{é}nfasis>. (Cited on page 71.)
- [9] *Class Robot*. URL <http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Robot.html>. (Cited on page 92.)
- [10] *VoxForge*. URL <http://www.voxforge.org/es>. (Cited on page 91.)
- [11] *Janus Recognition Toolkit (JRTk)*. URL http://en.wikipedia.org/wiki/Janus_Recognition_Toolkit_28JRTk29. (Cited on page 49.)
- [12] *La clase Robot en Java*. 2009. URL <http://jeandybryan.blogspot.com/2009/01/la-clase-robot-en-java.html>. (Cited on page 93.)
- [13] *Matriz Estocástica*. 2012. URL <http://es.scribd.com/doc/49785587/Matriz-Estocastica>. (Cited on page 19.)
- [14] *Revista Epistemowikia*, volume 6. de julio a septiembre del 2012. URL http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Reconocimiento_Automatico_del_Habla. (Cited on page 5.)
- [15] *Cadenas de Markov*. Universidad de Buenos Aires. Septiembre 12 2009. URL <http://operativa7107.awardspace.com/apuntes/MarkovParte1.pdf>. (Cited on page 17.)
- [16] ÁLVAREZ MARQUINA Agustín. *Historia de los sistemas de reconocimiento automático del habla*. Universidad Politécnica Madrid. 2010. URL http://tamarisco.datsi.fi.upm.es/ASIGNATURAS/FRAV/apuntes/historia_rv.pdf. (Cited on page 3.)

- [17] LEE Akinobu. *The Julius book*. 17 de Mayo 2010. URL <http://iiij.dl.sourceforge.jp/julius/47534/Juliusbook-4.1.5.pdf>. (Cited on page 83.)
- [18] CEBALLOS ARIAS Alexander. *Desarrollo de un sistema de manipulación de un robot a través de movimientos de la boca y de comandos de la voz*. 2009. URL <http://www.bdigital.unal.edu.co/3416/1/alexanderceballosarias.2009.pdf>. (Cited on page 5.)
- [19] AHUACTZIN LARIOS Angélica. *Diccionario español/Inglés para el aprendizaje de vocabulario utilizando una interfaz de voz*. Universidad de las Américas Puebla. Cholula, Puebla, México, 9 de diciembre de 1999. URL http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/ahuactzin_l_a/capitulo1.pdf. (Cited on page 3.)
- [20] CANO PANIZO María Cristina. *Módulo de habla CORBA para SOUL*. 2005. URL http://tierra.aslab.upm.es/documents/PFC/PFC_CPanizo.pdf. (Cited on page 47.)
- [21] ROBALINO PUENTE Luis Daniel. *Diseño e Implementación de un Control Remoto por Ordenes de la Voz para Aplicaciones de Control de una Vivienda*. Escuela Politécnica Nacional. Quito, Junio 2007. URL <http://bibdigital.epn.edu.ec/bitstream/15000/4167/1/CD-0886.pdf>. (Cited on pages 13 and 14.)
- [22] SAN MARTÍN Juan Eugenio. *Introducción al Análisis Espectral*. Facultad de Bellas Artes Universidad Nacional de la Plata. 12 de Marzo del 2010. URL http://www.astormastering.com.ar/Clase_1_Introducci~A^{3}n_al_analisis_espectral.pdf. (Cited on page 71.)
- [23] PEREZ VERDÚ Gonzalo. *Herramientas de Segmentación y Evaluación de Series Temporales Basadas en Modelos Ocultos*. 12 de octubre del 2010. (Cited on page 19.)
- [24] ROJO Horacio and MIRANDA Miguel. *Cadenas de Markov*. Universidad de Buenos Aires. Septiembre 12 2009. URL <http://operativa7107.awardspace.com/apuntes/MarkovParte1.pdf>. (Cited on page 17.)
- [25] VILLAMIL ESPINOSA Iván Horacio. *Aplicaciones en reconocimiento de la voz utilizando HTK*. Pontificia Universidad Javeriana, Santa fe de Bogotá. Mayo del 2005. URL <http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis95.pdf>. (Cited on pages xvi, xvii, 46, 59, 61, 62, 63, 64, 70, 74, and 76.)
- [26] SÁNCHEZ VALLINOT Jesús. *Módulo de transcripción voip-texto para una plataforma de Interceptación legal de comunicaciones*. Universidad Carlos III de Madrid. URL http://e-archivo.uc3m.es/bitstream/10016/14374/1/PFC_Jesus_Vallinot_Sanchez.pdf. (Cited on pages 46 and 49.)
- [27] CÁRDENAS MARISCAL Armando Juan. *Análisis de dos sistemas reconocedores de voz independiente y dependiente del contexto en cuanto precisión de reconocimiento, tiempo de entrenamiento y tiempo de pruebas*. Benemérita Universidad Autónoma de Puebla. 11 de febrero del 2005. (Cited on pages xvii and 48.)

- [28] SARMIENTO CARNEVALI María Laura. *Reconocimiento Automático del Habla Mediante un Modelo Híbrido basado en Modelos Ocultos de Markov y Máquinas de Vectores de Soporte*. Universidad de los Andes Mérida. Venezuela, Junio del 2008. URL http://tesis.ula.ve/pregrado/tde_busca/arquivo.php?codArquivo=1867. (Cited on pages xv and 24.)
- [29] MOYA GARCÍA Lisette. *Un etiquetador morfológico para el español de Cuba*. Universidad de Oriente Santiago de Cuba. Septiembre 2008. URL http://www.cerpamid.co.cu/sitio/files/Lisette_tesis_maestria.pdf. (Cited on page 23.)
- [30] VÁZQUEZ SÁNCHEZ Lucio. *Trazado de rutas de robots utilizando Modelos Ocultos de Markov*. Universidad Autónoma de Puebla. 30 de noviembre del 2011. URL <http://perseo.cs.buap.mx/bellatrix/tesis/TES1477.pdf>. (Cited on page 21.)
- [31] ARRIERO José María and ZARAPUZ Rosana. *Reconocimiento Digital del Habla*. Universidad Carlos III de Madrid. Leganés, 28 de Enero del 2009. URL <http://www.it.uc3m.es/jvillena/irc/practicas/06-07/02.pdf>. (Cited on page 5.)
- [32] DIAZARAQUE MARIN Juan. Miguel. *Cadenas de Markov*. Universidad Carlos III de Madrid. Madrid, 2004. URL <http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/PEst/tema4pe.pdf>. (Cited on pages 29, 30, 31, 33, 36, 37, and 38.)
- [33] AGUILERA BONET Pablo. *Reconocimiento de Voz Usando HTK*. Universidad de Sevilla. URL <http://bibing.us.es/proyectos/abreproy/11529/fichero/borradorHMM.pdf>. (Cited on pages xvi, xvii, 39, 41, 61, 62, 69, 71, 74, 76, 83, 89, and 91.)
- [34] CARRILLO AGUILAR Roberto. *Diseño y Manipulación de Modelos Ocultos de Markov Utilizando Herramientas Htk*. URL <http://www.scielo.cl/pdf/ingeniare/v15n1/Art03.pdf>. (Cited on pages xvii and 47.)
- [35] SOLERA UREÑA Rubén. *Máquinas de Vectores Soporte para Reconocimiento Robusto del Habla*. Universidad Carlos III de Madrid. Leganés, 2011. URL http://e-archivo.uc3m.es/bitstream/10016/12577/1/Tesis_Ruben_Solera_Urena.pdf. (Cited on page 3.)
- [36] GUTIERREZ NAVARRETE Tomas. *Detección de anomalías en la carga de un procesador, utilizando modelos ocultos de Markov*. Instituto Tecnológico de Morelia. Morelia, Michoacán, México. URL <http://www.asiat.com.mx/tomas/tesismaestria/micrositio/node2.html>. (Cited on pages xvii, 20, 21, and 26.)
- [37] L. u. J. a. Rodriguez-Fuentes, A. m. V. arona, G. e. B. ordel, I. Torres, K. L. de Ipiña, and J. M. Alcaide. *Reconocimiento automático del habla: Perspectiva desde un grupo de investigación del País Vasco*, pages 317 – 323. Congresos de Estudios Vascos. XIII Congreso de Estudios Vascos. Ciencia, tecnología y cambio social en Euskal Herria, Eusko Ikaskuntza, Donosti, 1996. URL <http://gtts.ehu.es/gtts/NT/fulltext/RodriguezEtal96.pdf>. (Cited on page 3.)

- [38] HERREROS Antonio VEGA Angel and RUBIO Antonio. *Reconocimiento Automático de Voz en Condiciones de Ruido*. Universidad de Granada. Granada. URL http://ceres.ugr.es/~atv/Documents/Docs/doctorado_atv/monograf_atv.pdf. (Cited on page 14.)
- [39] MARTINEZ Fernando y otros. *Reconocimiento de voz, apuntes de cátedra para Introducción a la Inteligencia Artificial*. URL http://www.secyt.frba.utn.edu.ar/gia/IA1_IntroReconocimientoVoz.pdf. (Cited on pages xvii, 16, and 19.)
- [40] YOUNG Steve y otros. *The HTK Book*, volume version 3.4 of *Cambridge University Engineering Department*. Diciembre del 2005. URL <http://htk.eng.cam.ac.uk/docs/docs.shtml>. (Cited on pages xvii, 62, 70, and 71.)
- [41] JIMÉNEZ Raúl y ROMERA Rosario. *Procesos Estocásticos*. 26 de febrero del 2009. URL <http://ocw.uc3m.es/estadistica/procesos-estocasticos-con-aplicaciones-al-ambito-empresarial/presentaciones/5CMcontinuas.pdf>. (Cited on page 37.)

Parte VI

ANEXOS

PROCESO INSTALACION DE HTK EN LA PLATAFORMA GNU/LINUX

Para trabajar con HTK es necesario instalar todos sus componentes de una manera correcta, pues existen ocasiones que se logran realizar aparentemente una instalación exitosa pero en el momento de correr los ejecutables de HTK existen errores comunes por falta de librerías o paquetes.

El proceso de instalación de esta herramienta es un paso importante que debemos efectuar, instalando cada paquete necesario.

Para comenzar a instalar HTK dentro de cualquier máquina recomendamos verifiquen lo siguiente:

- Micrófono en correcto funcionamiento.
- Probar un ejemplo de grabación de sonido (mediante el software que en su mayoría viene por defecto en cada sistema operativo).
- Parlantes en correcto funcionamiento (opcional, pues en las pruebas de reconocimiento es factible escuchar las grabaciones realizadas).

El manual fue realizado basado en las condiciones de software siguientes:

- Sistema operativo: Linux Mint (Debian) 32 bits
- HTK: 3.4.1

Paso 1. Descargar HTK

- En primer lugar debemos descargar HTK de la página oficial <http://htk.eng.cam.ac.uk/>, debemos registrarnos para poder hacerlo.

Nos descargamos htk en este caso la última versión existente es HTK 3.4.1

Podemos descargarnos los dos archivos el uno es HTK y el otro es un archivo que contiene ejemplos.

- HTK-3.4.1.tar.gz
- HTK-samples-3.4.1.tar.gz

Paso 2. Descomprimir archivos descargados

- Accedemos en el terminal a la ubicación en la que se encuentra nuestras descargas.
- Procedemos a desempaquetarlos mediante:
 - `tar xzf HTK-3.4.1.tar.gz`
 - `tar xzf HTK-samples-3.4.1.tar.gz`

Paso 3. Instalación de los paquetes necesarios

El siguiente paso es instalar paquetes necesarios para que la instalación de HTK se realice correcta y completamente.

- Como una recomendación antes de comenzar a instalar paquetes necesarios podemos realizar la siguiente sentencia:

sudo apt-get update

La cual nos brinda una actualización del sistema operativo, actualizando paquetes a versiones más recientes, o instalando paquetes necesarios para ciertas aplicaciones existentes.

Una vez actualizado nuestros paquetes, en primer lugar necesitamos tener instalado dentro de nuestro sistema operativo el lenguaje C++ puesto que este lenguaje es el que se encarga de compilar HTK para posteriormente crear e instalar los ejecutables de la herramienta.

(La mayoría de sistemas operativos Linux actuales ya poseen este paquete instalado).

Una vez actualizado nuestros paquetes, en primer lugar necesitamos tener instalado dentro de nuestro sistema operativo el lenguaje C++ puesto que este lenguaje es el que se encarga de compilar HTK para posteriormente crear e instalar los ejecutables de la herramienta. (La mayoría de sistemas operativos Linux actuales ya poseen este paquete instalado).

- Instalar C++ si no existe en nuestro sistema operativo.

sudo apt-get install g++

- O Build essential que se trata de una lista de paquetes que son esenciales para la creación de paquetes de Linux.

Con la siguiente sentencia:

sudo apt-get install build-essential

- De igual manera se necesita instalar los siguientes paquetes para continuar sin problemas,
 - libc6-i386
 - libc6-dev-i386
 - xorg-dev
 - libx11-xcb-dev
 - libx11-dev
 - ia32-libs

Como experiencia hemos verificado que se presentan ciertos inconvenientes en realizar la instalación de todos estos paquetes pues existen ya sus actualizaciones y no nos permite instalar el paquete que no sea de una versión actual.

Entonces si alguno de ellos no se instala continuamos con el resto de paquetes.

Estos son los paquetes que principalmente hemos verificado que son necesarios o que comúnmente pueden ocasionar ciertos inconvenientes

al momento de la instalación.

Ahora todo esto va a depender mucho de la distribución y versión del sistema operativo que se esté utilizando.

Paso 4. Compilación de HTK

En este proceso es donde interviene C para realizar la compilación de htk.

Una vez descomprimido los paquetes descargados de la página de HTK, ingresamos a la carpeta que se crea con los archivos descomprimidos.

cd htk

Una vez dentro de este directorio procedemos a realizar lo siguiente:

./configure --prefix=[directorio en el que se va a crear los ejecutables de HTK]

El directorio puede ser cualquiera, el que desee que obtenga los ejecutables a usar para la herramienta.

El directorio por defecto es: /usr/local

Recomendamos utilizar la misma dirección ya que no afecta en nada al rendimiento de HTK.

Es decir nuestra sentencia nos quedará de la siguiente manera:

./configure --prefix=/usr/local

Si realizamos de esta manera nuestros ejecutables se va a crear en la siguiente dirección:

/usr/local/bin

- Luego de ejecutar la sentencia del “configure”, realizamos el siguiente paso.

Para construir las bibliotecas de HTK y sus herramientas debemos ejecutar lo siguiente: (Para evitar inconvenientes de permisos vamos a ejecutar la sentencia make mediante sudo).

sudo make all

Y posteriormente:

sudo make install

Para instalarlos.

Y como mencionamos anteriormente se van a instalar en el directorio que se ingreso es decir en nuestro caso: /usr/local/bin.

Paso 5. Agregar nuestro directorio al PATH¹ de nuestro sistema

(Este paso solo se realizará si en el Paso A se escogió un directorio diferente al que viene por defecto, o de lo contrario continuamos al Paso A)

Debemos tomar en cuenta algo muy importante si se escoge una dirección distinta y según su criterio debemos agregar el directorio que se escogió al PATH del sistema para tener las herramientas y variables de HTK siempre disponibles desde cualquier ubicación.

La manera de agregar nuestro nuevo directorio al Path es la siguiente:

(Todos estos pasos dependerá de la distribución de Linux con la que se esté trabajando)

- Tenemos que realizar las siguientes modificaciones en el archivo

/etc/profile

- Nos creamos una nueva variable con el directorio escogido

NUEVAVARIABLE=/usr/local/bin

- Ubicamos la línea que tiene el comando “export” y agregamos al final de la línea la variable que creamos anteriormente (NUEVAVARIABLE).

Entonces nos quedaría de la siguiente manera:

**export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE
INPUTRC NUEVAVARIABLE**

Guardamos los cambios realizados en el archivo /etc/profile , y listo, podemos usar las herramientas de HTK desde nuestra consola.

Paso 6. Prueba de funcionamiento

Ahora, por ejemplo vamos a convocar a una herramienta de HTK llamada HSLab² y posteriormente según lo que se necesite realizar con HTK

¹ PATH. Una ruta (en inglés path) es la forma de referenciar un archivo o un directorio, en la que se especifican las rutas en las cuales el intérprete de comandos debe buscar los programas a ejecutar

² HSLab- Herramienta grafica de HTK que se la utiliza para realizar grabaciones de voz y etiquetar las mismas.

llamamos a la función que necesitemos.

Digitamos en la consola:

HSLab nombre.sig

Obteniendo el siguiente resultado:

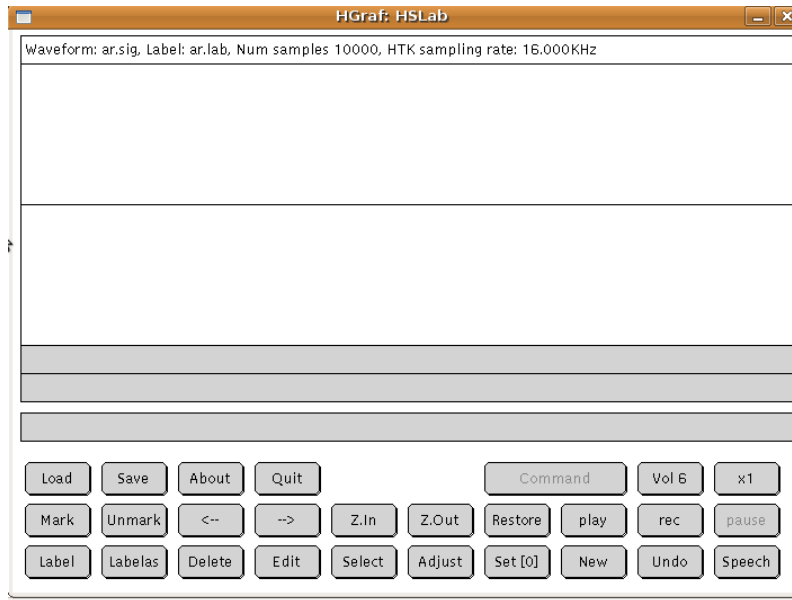


Figura 53: Interfaz de HSLab

Procedemos a grabar una muestra de voz con el botón “rec” y luego la reproducimos, si no nos presenta ningún inconveniente, entonces todas las herramientas de HTK funcionan correctamente.

PRUEBAS Y PROBLEMAS

En el momento de implementar los pasos de instalación tuvimos una serie de inconvenientes, pues hicimos las pruebas en varias versiones principalmente en Ubuntu.

Ubuntu 8.10, 9.10, 10.4, 10.10, 11.4, 11.10, kubuntu 11.10 y fedora 16.

Como se mencionó anteriormente va a depender de cada versión pues cada una viene con cierta cantidad y tipo de paquetes y aplicaciones instaladas, entonces vamos a requerir instalar los paquetes que no vengan por defecto.

También va a depender del hardware disponible, pues drivers y aplicaciones no siempre son compatibles.

Al momento de trabajar con GNU/Linux puede haber la necesidad de activar drivers privativos o instalarlos si así es necesario para un buen trabajo.

PROBLEMAS ENCONTRADOS:

- No existe un reconocimiento del hardware (micrófono, parlantes).
- Aparentemente se realiza una instalación exitosa de HTK (sus ejecutables), pero al momento de probar presenta errores al momento de su ejecución.
- Problemas de audio, es decir, se realizó una instalación incorrecta o incompatible.
- Problema de configuración de sonido.
- Hardware incompatible que presenta una producción de ruido que puede afectar al uso adecuado de HTK.

ANEXOS

PROCESO DE INSTALACIÓN DE JULIUS

El proceso de instalación de Julius es sencillo, simplemente debemos compilar los archivos binarios de la herramienta y tendremos Julius instalado.

Para una instalación correcta de Julius debemos seguir los siguientes pasos:

Paso 1

Descargar el software de la página web oficial http://julius.sourceforge.jp/en_index.php

Hemos descargado la última versión existente, Julius 4.2.1.

Paso 2

Extraer el contenido del archivo descargado en un directorio específico.

Paso 3

Desde el terminal de Linux nos dirigimos a la ubicación del directorio en el cual se ha realizado la extracción del archivo.

Paso 4

Estos pasos son similares al proceso de compilación de HTK.

Corremos las siguientes líneas de código en el terminal para realizar la compilación:

```
./configure
```

```
sudo make
```

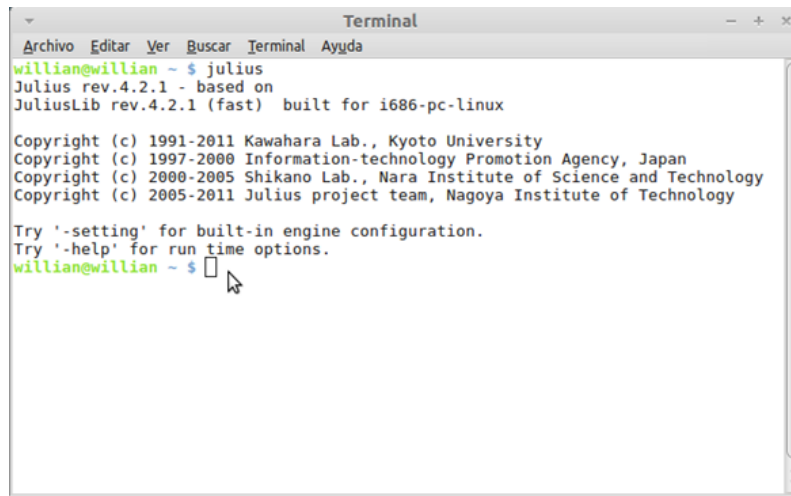
Paso 5

Instalamos los binarios compilados con la siguiente línea:

```
sudo make install
```

Y de este modo ya tendremos instalado Julius y podremos tener acceso a esta herramienta desde cualquier ubicación.

Para comprobar si se ha realizado la instalación con éxito, escribimos “Julius” en el terminal de Linux y debemos obtener el siguiente resultado:

A screenshot of a Linux terminal window titled "Terminal". The window has a menu bar with "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The terminal shows the command "julius" being executed. The output displays the version "Julius rev.4.2.1 - based on JuliusLib rev.4.2.1 (fast) built for i686-pc-linux", followed by copyright information for Kawahara Lab., Information-technology Promotion Agency, Shikano Lab., and the Julius project team. It also provides instructions to use "-setting" for engine configuration and "-help" for runtime options. The prompt "willian@willian ~ \$" is visible at the bottom with a cursor.

```
willian@willian ~ $ julius
Julius rev.4.2.1 - based on
JuliusLib rev.4.2.1 (fast) built for i686-pc-linux

Copyright (c) 1991-2011 Kawahara Lab., Kyoto University
Copyright (c) 1997-2000 Information-technology Promotion Agency, Japan
Copyright (c) 2000-2005 Shikano Lab., Nara Institute of Science and Technology
Copyright (c) 2005-2011 Julius project team, Nagoya Institute of Technology

Try '-setting' for built-in engine configuration.
Try '-help' for run time options.
willian@willian ~ $
```

Figura 54: Julius Información

Todo este proceso y más detalles se pueden encontrar en la documentación de Julius que se encuentra dentro del mismo sitio oficial de Julius del cual se descargó el software.