DISEÑO Y MANIPULACIÓN DE MODELOS OCULTOS DE MARKOV, UTILIZANDO HERRAMIENTAS HTK. UNA TUTORÍA

DESIGN AND MANIPULATION OF HIDDEN MARKOV MODELS USING HTK TOOLS. A TUTORIAL

Roberto Carrillo Aguilar[1]

Recibido 20 de abril de 2006, aceptado 8 de enero de 2007
Received: April 20, 2006 Accepted: January 8, 2007

RESUMEN

Este trabajo da a conocer el sistema de desarrollo de software para el diseño y manipulación de modelos ocultos de Markov, denominado HTK. Actualmente, la técnica de modelos ocultos de Markov es la herramienta más efectiva para implementar sistemas reconocedores del habla. HTK está orientado principalmente a ese aspecto. Su arquitectura es robusta y autosuficiente. Permite: La entrada lógica y natural desde un micrófono, dispone de módulos para la conversión A/D, preprocesado y parametrización de la información, posee herramientas para definir y manipular modelos ocultos de Markov, tiene librerías para entrenamiento y manipulación de los modelos ocultos de Markov ya definidos, considera funciones para definir la gramática y, además, una serie de herramientas adicionales para lograr el objetivo final de obtener una hipotética transcripción del habla (conversión voz-texto).

Palabras clave: Reconocimiento automático del habla, HTK, HMM.

ABSTRACT

This paper presents HTK, a software development platform for the design and management of Hidden Markov Models. Nowadays, the Hidden Markov Models technique is the more effective one to implement voice recognition systems. HTK is mainly oriented to this application. Its architecture is robust and self-sufficient. It allows a natural input from a microphone, it has modules for A/D conversion, it allows pre-processing and parameterization of information, it possesses tools to define and manage the Hidden Markov Models, libraries for training and use the already defined Hidden Markov Models. It has functions to define the grammar and it has additional tools to reach the final objective: to obtain an hypothetical transcription of the talking (voice to text translation).

Keywords: Automatic Speech Recognition, HTK, HMM.

INTRODUCCIÓN

Los sistemas más exitosos en el área de reconocimiento automático del habla (ASR: Automatic Speech Recognition), se basan en la utilización de la técnica de análisis estocástico: Modelos Ocultos de Markov (HMM: Hidden Markov Models) [3, 8]. La implementación de un sistema ASR no es una tarea cómoda para los investigadores, debido principalmente al alto grado de complejidad que alcanza este tipo de problema [3-5]. El principal obstáculo se refiere al manejo de grandes bases de datos (archivos, parámetros, modelos, diccionarios). Sin una herramienta computacional efectiva, los objetivos serán limitados, y hasta obstaculizadores para continuar con el desarrollo del estado del arte.

HTK, Hidden Markov Model Toolkit [1, 2 y 4]. Es un conjunto de herramientas de software para diseñar y manipular HMM. Originalmente fue creado para aplicarlo al desarrollo de sistemas ASR. Ahora puede utilizarse en cualquier área del conocimiento, la única restricción es que el problema a resolver pueda ser enfocado como un proceso de modelación Estocástico Markoviano. En la actualidad es exitosamente utilizado en: Reconocimiento y síntesis de voz, reconocimiento de caracteres y formas gráficas, análisis de vibraciones mecánicas. Incluso ha

[1] Universidad de La Frontera. Av. Fco. Salazar 01145. Temuco, Chile. rcarrill@ufro.cl

sido usado con éxito en determinar secuencias válidas del ADN humano (Proyecto Genoma). Según sea el grado de complejidad de nuestro problema (nivel al que se diseñen los HMM), HTK resulta adaptable al tipo y formato de dato ha utilizar y permite el diseño de diferentes tipos de reconocedores.

El desarrollo de HTK lo lleva a cabo el grupo del habla, visión y robótica del Departamento de Ingeniería de la Universidad de Cambridge (CUED), UK. Actualmente HTK es de libre distribución y su código y librería pueden ser modificados en común acuerdo con el CUED. Además la herramienta se encuentra disponible para utilizarlo en diversas plataformas o sistemas operativos, tales como: Unix, Linux, Windows XP y DOS.

FUNCIONAMIENTO DE HTK

HTK dispone de una arquitectura flexible y autosuficiente. Es controlado por módulos de librerías, que alimentan la interfaz de funciones correspondiente: Manejo de archivos, operaciones matemáticas, interacción con sistema operativo. La utilización de cualquier herramienta disponible depende de dos aspectos:

– Línea de comandos como interfaz con el sistema operativo. Dispone de parámetros opcionales para controlar detalles particulares en su ejecución [1, 2 y 4].
– Módulos de librerías: Hay librerías para el manejo de distintos tipos de archivos. Será común utilizar archivos de configuración para adaptarse al manejo de los módulos.

Uso de las herramientas

El volumen de archivos que HTK debe manejar y controlar es siempre elevado. La utilización de comandos es fundamental. Sólo así será práctico de utilizar. La línea de llamada a comandos es como sigue:

herramienta [opciones] archivos

[opciones]: Agrupa las variables, parámetros del sistema y librería. Las opciones particulares son indicadas mediante un guión, seguido de una letra, y acompañada del valor alfanumérico correspondiente:

Opción valor

Las letras minúsculas indican opciones propias de la herramienta, y el parámetro utilizado podrá tener diferentes significados, según la herramienta a ejecutar. Las letras mayúsculas son utilizadas para indicar opciones comunes a todas las herramientas del sistema, por ejemplo:

Opción –C: Se refiere a que el archivo indicado posee una lista de parámetros que deben ser actualizados.
Opción –S: Sirve para potenciar y flexibilizar el manejo de base de datos (archivos). Por ejemplo: Indicará a la herramienta que el archivo mencionado hace referencia a una lista de archivos (Muy útil en etapas de entrenamiento).
Opción –A: Imprime la línea de comandos con sus argumentos.
Opción –D: Muestra por pantalla las variables de configuración que se están utilizando.
Opción –T: Permite que durante la ejecución de la herramienta aparezca información relativa a la ejecución de la misma (uso de flags).
Opción –L: Busca las etiquetas en el directorio especificado.
Opción –H: Carga el HMM como un archivo macro de extensión mmf (Master Macro File).

La sintaxis es la usual para cualquier lenguaje de programación:

Parámetro = valor

Los valores para los parámetros pueden ser:

- Números enteros o flotantes.
- String.
- Booleanos.

Acepta cualquier formato usado en lenguaje C.

Librerías

Cada librería es un conjunto de instrucciones para lograr una función específica en las herramientas disponibles. Hay librerías comunes a todos los comandos y otras son específicas. Los principales módulos de librerías disponibles son las siguientes:

HAudio: Controla adquisición en vivo de señales de audio.
HAdapt: Adaptación del reconocedor a uno o más locutores.
HDict: Control del diccionario del reconocedor.
HGraf: Manejo gráfico de señales de audio.
HLabel: Manejo de archivos de etiquetas.
HLM: Maneja modelos de lenguajes.
HMath: Maneja estructuras de alto nivel tales como: Matrices y vectores.
HMem: Módulo para manejo de memoria de bajo nivel.
HModel: Interpreta las definiciones de HMM.
HNet: Soporta archivos con formato Lattice y Networks.

HParm: Control de parametrización en señales de audio.
HRec: Conjunto de funciones para el procesado en etapa de reconocimiento.

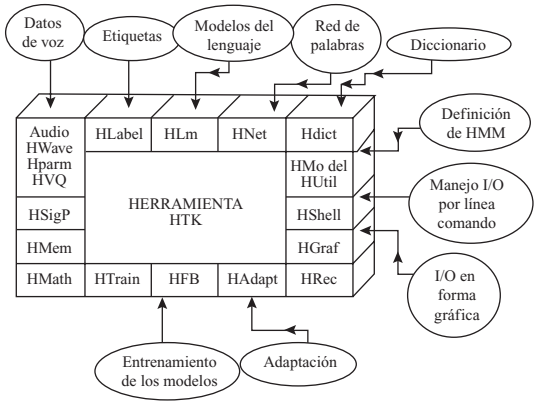

Figura 1. Arquitectura del entorno HTK.

HSigP: Ofrece operaciones para controlar alocuciones.
HShell: Es la interfaz entre HTK y el sistema operativo.
HTrain: Soporte para entrenamiento de modelos.
HUtil: Rutinas de manipulación de HMM.
HVQ: Manejo de codebook de vectores cuantificados.
HWave: Manejo de archivos entrada y salida.

PARAMETRIZACIÓN CON HTK

Dada una señal, debemos extraer su información. En el área de procesamiento de señales, al referirnos a lo anterior, hacemos referencia a la extracción de parámetros, los cuales nos servirán para entrenar los HMM y lógicamente efectuar el reconocimiento.

Formatos de Parametrización

HTK acepta dos formatos de parametrización:

– Formato de parámetros HTK: Opción por defecto.
– Formato de parámetros Esignal: Similar al formato HTK, sólo presenta diferencias en los cabezales de los archivos.

Existen tres posibilidades para la obtención de las muestras:

1. Archivos Parametrizados: Al experimentar con señales, lo primero que debemos lograr es disponer de una base de datos parametrizada, sólo así, posteriormente, podremos realizar múltiples experimentos. Para construirla, HTK dispone de la herramienta HCopy. Las entradas de HCopy son la señal en banda base (en formato wav) y un archivo de configuración de la parametrización, el cual le indica a HTK la manera de analizar los datos. HCopy entrega como salida un archivo de voz parametrizada convertido en vector.
2. Archivo de Señal: Para cualquier señal deberán tomarse las muestras y parametrizarlas. Sólo así, HTK podrá interpretarlas.
3. Entrada Directa por Micrófono: Es especial para experiencias con voz en vivo. Habrá que tener en cuenta el efecto del ruido ambiental y obligatoriamente se debe filtrar.

Parametrización de la señal

Como ya se indicó, HTK soporta todo el proceso de parametrización por intermedio del comando HCopy. Para hacerlo operativo se debe crear un archivo de configuración. Este archivo puede ser un archivo de texto codificado en ANSI; En él se debe indicar: Tipo de algoritmo en aplicar: Banco de Filtros, Cepstrum y MEL Cepstrum. Los parámetros para especificar el archivo de configuración son:

– Formato de datos: SOURCEFORMAT=HTK
– Tipo de algoritmo y análisis: En tal caso tenemos las siguientes posibilidades:

Tabla 1. Algunos algoritmos y tipos de análisis disponibles.

LPC	Predicción Lineal
LPREFC	Coeficientes de Reflexión para LPC
LPCEPSTRA	LPC con Cepstrum
LPDELCEP	LPC Cepstrum - Plus de Coeficientes
IREFC	LPC - Coeficientes de Reflexión
MFCC	Coeficientes MEL Cepstrum
FBANK	Banco de filtros de MEL con salida LOG
MELSPEC	Banco de filtros de MEL con salida Lineal

También se deben indicar parámetros, tales como:

– Ventana de análisis: WINDOWSIZE (tamaño).
– Duración de las muestras: TARGETRATE.
– Número de coeficientes cepstrum: NUMCEPS.
– Filtro de Preénfasis: PREEMCOEF.
– Número de bancos de filtros: NUMCHAMS.
– Ganancia Cepstral: CEPLIFTER.

Y un grupo finito de coeficientes extras (muy útiles para afinar el análisis):

– Cantidad de Energía: _E
– Energía total Suprimida: _N
– Coeficientes Delta: _D

– Coeficientes de Aceleración: _A
– Uso de compresión: _C
– Valor medio nulo: _Z
– Coeficiente Cepstral nulo: _0

Ejemplo, por intermedio de la variable TARGETKIND= MFCC_E_D_A; HTK sabrá del tipo de análisis (significa que el algoritmo es un banco de filtros con coeficientes cepstrales MFCC, al que se le añade la energía _E, los coeficientes Delta _D y los de aceleración _A). Un ejemplo de archivo de configuración de la parametrización se muestra a continuación. El tamaño de la ventana para el análisis es de 25 ms y el tiempo para cada muestra alcanza a los 10 ms, los restantes parámetros ya fueron explicados.

```
SOURCEFORMAT=HTK
TARGETKIND=MFCC_0_D_A
WINDOWSIZE=250000
TARGETRATE=100000
NUMCEP=12
USERHAMMING=T
PREEMCOEF=0.97
NUMCHANS=26
CEPLIFTER=22
```

Figura 2. Archivo de configuración de la parametrización.

La llamada de interés para ejecutar HCopy es:

HCopy [opción] archivo parametrización [opción] archivo salidaParametrizada.

En contados casos, el llamado a una herramienta es sencillo. Para hacer más fácil la comprensión en el uso de cada herramienta, se utilizará, en adelante: Diagramas en bloque.

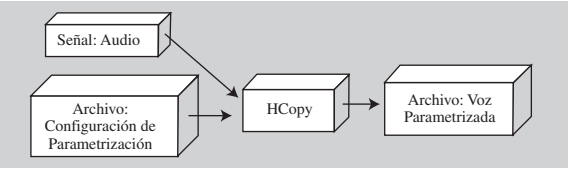

Figura 3. Esquema en bloques de la llamada a HCopy.

El archivo salidaParametrizada debe poseer extensión mfcc, indicando con ello que los datos corresponden a vectores codificados según coeficientes cepstrales de frecuencia MEL.

Para visualizar los resultados del proceso de parametrización, HTK dispone del comando HList. Si el usuario desea conocer el vector de parametrización para alguna forma de onda en particular, la llamada debe ser:

HList [opción] archivo de audio

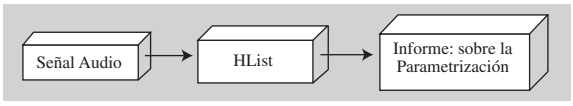

Figura 4. Diagrama en bloque de herramienta HList.

La entrada: Archivo de audio es la señal de audio en formato wav. HList entrega un archivo con el informe de la parametrización de las muestras.

Análogamente, HTK dispone de recursos para realizar la cuantización de los vectores (o etiquetado). Se puede optar por los modelos de densidad discreta o continua, en estos últimos los vectores cuantizados se agrupan como un codebook [3], lo cual permite acelerar el proceso de reconocimiento reduciendo el tiempo de cálculos, pero reduce la precisión debido a que fácilmente aumentará la tasa de error, dando lugar a reconocedores débiles. La herramienta HSlab es la encargada de realizar el etiquetado en HTK. Se dispone para ello de un ambiente muy amable (completamente gráfico), lo cual simplifica el proceso.

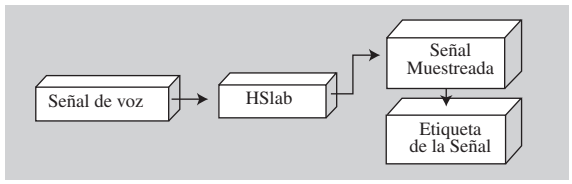

Figura 5. Esquema en bloque de la llamada a HSlab.

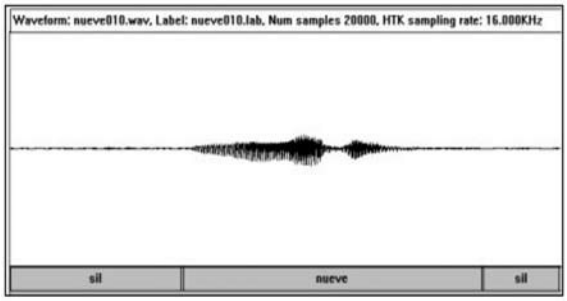

Figura 6. Etiquetado de una señal de audio.

CREACIÓN DE MODELOS DE MARKOV

El concepto de HMM está ampliamente difundido, siendo el estudio [3] el pilar en su aplicación en sistemas ASR.

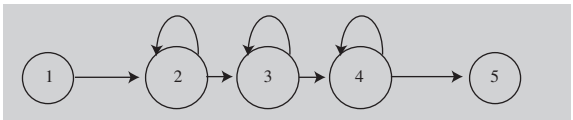

Figura 7. Esquema de un HMM.

Los elementos característicos de todo HMM, son:

- Número de estados: N
- Número de símbolos observables por estado: M.
- Probabilidad de transición entre estados: $A=\{a_{ij}\}$.
- Distribución de Probabilidad de símbolos de observación: $B=b_j(k)$.
- Distribución inicial de estados: $\Pi=\{\pi_i\}$.

Para que un HMM sea útil [3], deben responderse las tres siguientes preguntas:

1.- ¿Para el modelo, cómo se evalúa la probabilidad de la secuencia observada?

El algoritmo Fordward – Backward [3, 4 y 7] permite resolver este problema. Consiste básicamente en un proceso que permite optimizar el cálculo de la probabilidad, de lo contrario, el procesamiento (multiplicaciones y adiciones) sería tan grande que no podría implementarse de manera práctica. Con su aplicación, bastan $2TN^2$ operaciones de cómputo, en vez de $2TN^T$ operaciones al no utilizar el algoritmo. Siendo T, el número de observaciones realizadas.

2.- ¿Cómo elegir la secuencia óptima de estados?

Este problema se puede resolver abordando la técnica de programación dinámica, específicamente utilizando el algoritmo de Viterbi [3, 4 y 6].

3.- ¿Cómo ajustar los parámetros del modelo?

Basándonos en algún criterio de optimización podríamos responder a esta pregunta, y para ello es necesario utilizar el algoritmo de Baum – Welch [3, 4 y 6].

Implementación de un HMM

Su implementación se realiza mediante un editor de textos cualquiera. El objetivo es diseñar un único prototipo de HMM para especificar las características generales de todos los modelos que más adelante formen parte del sistema. Deben indicarse los siguientes parámetros:

- Tipo de vector de observación.
- Número de estados.
- Número de flujos de datos (data streams) y su ancho.
- Cada estado emisor (de cada flujo) se indica con el número de componentes mixtas (mixtures component) de la gaussiana y su correspondiente peso. Los parámetros de cada componente son:
  - Media y covarianza.
  - Parámetros opcionales (duración del vector y peso del flujo).
  - Matriz de transición.

El modelo más sencillo será aquel que no tiene ningún parámetro global, es decir, que no comparte ninguna característica de las anteriormente nombradas con ningún otro modelo. La complejidad de los modelos puede ir aumentando según se vayan compartiendo más características (requiere uso de macros). Una macro es cualquiera de los grupos de parámetros del modelo almacenado externamente al mismo. Así, para hacer uso en un modelo de un parámetro definido mediante una macro, sólo habrá que incluir una línea en la que se llame a esa macro.

El lenguaje de definición de modelos utiliza el mismo carácter clave para indicar las macro de los distintos parámetros (*~tipo de macro)*. Sin embargo, se pueden distinguir dos tipos de macro (según su uso):

- Macro para un modelo: Los parámetros internos de un modelo concreto se escriben como unidades separadas (como archivos diferentes). Se pueden hacer macro de todos los parámetros indicados en un modelo. Las macro más empleadas son las de los parámetros correspondientes a cada uno de los estados. De este modo se puede tener, por ejemplo, la misma varianza en todos los estados de un modelo.
- Macros comunes para varios modelos: Idéntico al anterior, pero habrá varios modelos compartiendo la misma macro. Se lograría, por ejemplo, que todos los modelos tengan algún estado con un mismo vector de varianza.

Ampliamente se ha discutido sobre cómo generar un prototipo de HMM [3, 8], llegándose a la siguiente conclusión: Un modelo con parámetros aislados es más sencillo de realizar, pero querer lograr una elevada tasa de aciertos en el reconocedor requerirá enlazar estados o modelos. La opción más útil parece ser: partir de un modelo sencillo, donde se encuentren todos los parámetros que se van a manejar de modo aislado.

Lenguaje para definición de modelos

HTK usa un lenguaje propio para definir los modelos, así todas las herramientas funcionarán correctamente con el mismo HMM. Conviene conocer este lenguaje para poder editar un prototipo acorde con nuestras necesidades.

El ejemplo en la figura 8 corresponde a un modelo de cinco estados, tres son de emisión (los estados de entrada

y salida no emiten. Ver figura 7). La primera línea indica el nombre del HMM, *~h "nombre"*. Aquí se está definiendo el modelo llamado "hmm0". En la siguiente línea hay que poner la palabra clave *<BeginHMM>*, que indica el punto a partir del cual se encuentran los parámetros del modelo. Al final de la definición habrá otra palabra clave complementaria *<EndHMM>*.

```
~h "hmm0"
<BeginHMM>
        <VecSize>4<MFCC>
        <NumState>5
        <State>2
                <Mean>4
                0.2 0.1  0.1 0.9
                <Variance>4
                1.0   1.0  1.0  1.0
        <State>3
                <Mean>4
                0.4 0.9 0.2 0.1
                <Variance>4
                1.0   2.0  2.0  0.5
        <State>4
                <Mean>4
                1.2 3.1 0.5 0.9
                <Variance>4
                5.0 5.0 5.0 5.0
        <TransP>
        0.0  0.5  0.5  0.0  0.0
        0.0  0.4  0.4  0.2  0.0
        0.0  0.0  0.6  0.4  0.0
        0.0  0.0  0.0  0.7  0.3
        0.0  0.0  0.0  0.0  0.0
<EndHMM>
```

Figura 8. Definición de un prototipo de HMM.

El orden dentro de la jerarquía es de mayor a menor importancia:

- <BeginHMM>, <EndHMM>
- Parámetros globales, número de estados, definición de cada estado y matriz de transición entre estados.
- Número de flujos de dato y sus pesos.
- Número de componentes mixtas y sus pesos.
- Media y varianza de cada uno de las componentes mixtas.

Al principio se especifican los aspectos globales de todos los modelos que conformen la base, los que suelen indicarse son: Tamaño de los vectores de parámetros *<VecSize> número entero* y el algoritmo con el que se han extraído la información de la forma de onda. El algoritmo se pone entre llaves, como el resto de palabras clave. En este caso *<MFCC>*. Para no dar lugar a errores, el nombre del algoritmo será el indicado durante el proceso de parametrización. En la siguiente línea se deben indicar el número de estados que componen el modelo. A partir de este punto se encuentra la definición de cada estado. La información recogida para cada estado depende de cuántos flujos de datos se utilizan y del número de componentes de mezcla gaussiana en cada flujo. Existen también casos que mezclan varios flujos de datos y cada uno de ellos cuenta con diversas componentes de mezcla. Una vez alcanzada la componente de mezcla gaussiana como unidad fundamental, los parámetros serán:

- Vector de valores medios.
- Vector de varianza, en alguno de los siguientes formatos:
  - Varianza. El vector es la diagonal de la matriz.
  - Covarianza. La distribución gaussiana se indica como una matriz completa. Estas son simétricas, por lo que sólo se almacena la diagonal superior. También podemos usar la descomposición de Choleski de la matriz inversa de la covarianza, y almacenando la matriz triangular superior.

Una vez que se han definido todas las componentes de mezcla de todos los flujos y para todos los estados, el prototipo de HMM se completa con la matriz de transición de estados *<TransP> tamaño*. Esta debe ser una matriz cuadrada, cuyo tamaño hay que indicar siempre de modo explícito, y debe corresponder con el número de estados definidos anteriormente. La suma de todas las componentes de cada fila debe resultar la unidad, excepto para el último estado, cuya suma debe ser siempre cero (no se permite ninguna transición que parta del estado final hacia alguno de los estados anteriores). Finalmente el modelo se termina con <EndHMM>.

ENTRENAMIENTO DE LOS HMM

Básicamente, consiste en actualizar los valores de los parámetros mediante diferentes algoritmos y, con ello, conseguir modelos más ajustados a la realidad o aplicación. Por lo tanto, debe ser un proceso optimizado. Consta de dos fases: i) Inicialización, que se consigue por medio de la herramienta HInit. ii) Aplicación del algoritmo de Baum-Welch.

También está disponible la posibilidad de realizar un entrenamiento global, lo cual siempre se debería realizar. HTK dispone de la herramienta HERest para dicha tarea. Veamos las herramientas para el entrenamiento:

HInit aprovecha las facilidades del segmentado de etiquetas para detectar los diferentes alófonos. Lo buscará en cada modelo. Posteriormente, utiliza el algoritmo de Viterbi, para la estimación de las medias y varianzas. El proceso se repite, hasta que el algoritmo ya no pueda mejorar la

semejanza, la cual es calculada paralelamente. La llamada para inicializar cada modelo de cada alófono se hace de modo iterativo, su formato se muestra en la figura 9.

HInit [opción] datos_parametrizados [opción] directorio_ destino [opción] \ archivo_fuente [opción] etiqueta [opción] directorio_fuente_etiquetas

HRest utiliza el algoritmo de Baum - Welch (reestimación de los parámetros de los modelos aislados). Este modo de entrenamiento debe realizarse de modo iterativo, para cada alófono del conjunto. Las opciones de esta herramienta son poderosas [1, 4]. Su diagrama de llamada se muestra en la figura 10.

HERest realiza un entrenamiento global actualizando simultáneamente todos los modelos del sistema. Se ignora la información de segmentado en las etiquetas, pues sólo interesa conocer el mensaje que contiene la frase. Cada etiqueta usada en este entrenamiento acumula estadísticas provenientes del algoritmo Forward - Backward. Su diagrama de llamada es mostrado en la figura 11.

Respecto del significado de cada archivo en fase de entrenamiento; las llamadas a cada herramienta son:

Datos_parametrizados: Son los archivos que conforma la base de datos parametrizada (archivos con extensión mfcc), a entrenar. Generalmente se indica bajo opción S (script), a la manera de cómo acceder a esos datos.
Archivos_salida_Estadística: Es el archivo que genera el proceso HERest, como resultado estadístico del análisis del entrenamiento.
Directorio_fuente_etiquetas: Es la ruta para alcanzar el archivo que contienen la lista con todas las etiquetas del proceso.
Lista_archivos_modelos_ocultos_de_markov: Es la ruta para ubicar al archivo con todos los modelos de Markov actualizados.
Lista_nombre_sonidos_a_reconocer: Es la indicación que caracteriza al archivo de entrada que contiene la lista de todo los HMM del proceso.

RED GRAMATICAL Y DICCIONARIO

Red gramatical es la secuencia de palabras que pueden reconocerse, en cambio, Diccionario describe los diferentes alófonos que componen cada una de las palabras (red). Pueden crearse distintos tipos de redes, dependiendo del modelo de lenguaje utilizado. Si el problema consiste en reconocer señales de voz con nombre y apellido, entonces un ejemplo de red gramatical será la mostrada en la figura 12. La sintaxis divide la red en tres partes: cabecera, nodos y enlaces. Los enlaces pueden ser: unigrama (probabilidad de una palabra) o bigrama (transiciones posibles entre unas palabras hacia otras).

Los valores de probabilidad de los enlaces dependen del modelo de lenguaje usado. Se definen tres tipos: i) Wordloop: Todos equiprobables. ii) Bigrama: Cada enlace tendrá una probabilidad que se ajusta al número de ocurrencias de esa transición en el texto de entrada. iii) Modelo propio: No asignan probabilidades, indica cuáles son los enlaces permitidos. Esta estructura resulta ideal cuando se usan diccionarios con muy pocas palabras.

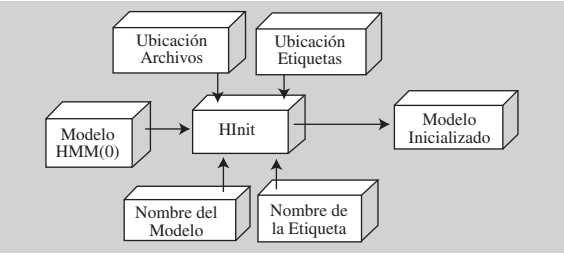

Figura 9. Diagrama en bloque para llamar a HInit.

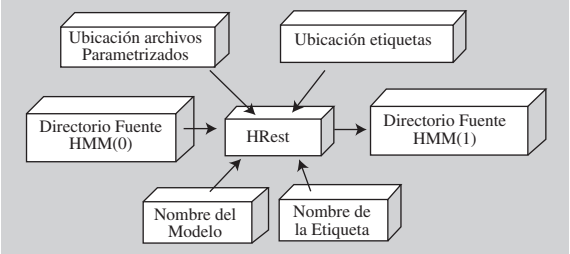

Figura 10. Diagrama en bloque para llamar a HRest.

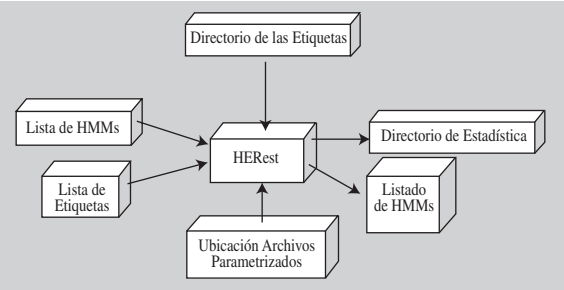

Figura 11. Diagrama en bloque para llamar a HERest.

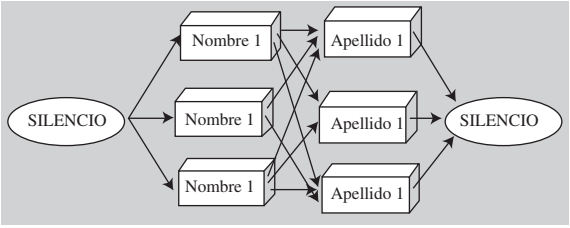

Figura 12. Estructura de una red gramatical.

Herramientas para el manejo de la red gramatical

Tres son las herramientas para esta opción: HLStats, HBuild y HParse.

HLStat: Realiza las estadísticas necesarias para la creación del bigrama. Determina las probabilidades de aparición de las palabras del diccionario. Se emplea la siguiente fórmula:

$$p(i,j) = \begin{cases} \dfrac{N(i,j) - D}{N(i)} & \forall\ N(i,j) > t \\ \\ b(i)\,p(i) & \forall\ N(i,j) \le t \end{cases}$$

donde:

$N(i, j)$ Número de veces que j está después de i.
N(i) Número de apariciones de i.
D Constante auxiliar (valor típico 0.5).

Para unigrama, la fórmula empleada es:

$$p(i,j) = \begin{cases} \dfrac{N(i)}{N} & \forall\ N(i,j) > t \\ \\ \dfrac{u}{N} & \forall\ N(i,j) \le t \end{cases}$$

u Base del unigrama. Por defecto vale 1.

HBuild: Se encarga de generar las redes partiendo de distintos valores de probabilidad. Genera salidas cuando el modelo de lenguaje utilizado es: Wordloop o bigrama. Utiliza una lista de palabras, y no el diccionario, pues no necesita descomponer las palabras en alófonos. En este modelo de lenguaje las palabras son equiprobables.

HParse: En los casos anteriores se creaban todos los enlaces posibles, ahora con HParse, las transiciones de palabras estarán sujetas a la sintaxis elegida por el propio usuario, pues es una herramienta utilizable sólo cuando el usuario crea un modelo de lenguaje propio. Por ejemplo, para reconocer nombres y apellidos. Un ejemplo de su entrada típica, para reconocer nombres y apellidos, será:

HParse [opción] Archivo_de_gramática [opción] red_ palabras_de_salida. Ver figura 13.

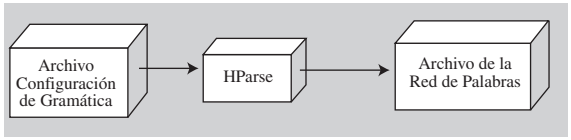

Figura 13. Diagrama en bloque para llamar a HParse.

RECONOCIMIENTO Y EVALUACIÓN DE RESULTADOS

Una vez configurada la red gramatical debemos ingresar a la etapa de reconocimiento y evaluación de resultados. Esta opción sólo requiere de archivos parametrizados y de un repertorio de alocuciones (base de datos para el reconocimiento). Los resultados los da a conocer un archivo de salida, siendo fundamental el porcentaje de reconocimiento.

Archivos MLF: En el proceso de reconocimiento se generan etiquetas de los distintos archivos que se han utilizado. Cuantos más archivos se reconozcan, más fiable podrá ser la estadística de aciertos que se calcule. La manera natural de almacenar las etiquetas es utilizando un archivo de etiquetas maestro (Master Label File: MLF). El cual cumple una función idéntica a las macro de los HMM; almacenando dentro de un único archivo todas las etiquetas (de allí su nombre).

HVite: Es la herramienta que maneja el algoritmo de Viterbi. Debe proporcionar el camino de mayor probabilidad para la secuencia óptima de estados. Mientras más grande es la red, más procesamiento requerirá el algoritmo y, por lo tanto, más lento resultará el reconocimiento. Su llamada la indica el siguiente diagrama en bloques de la figura 14.

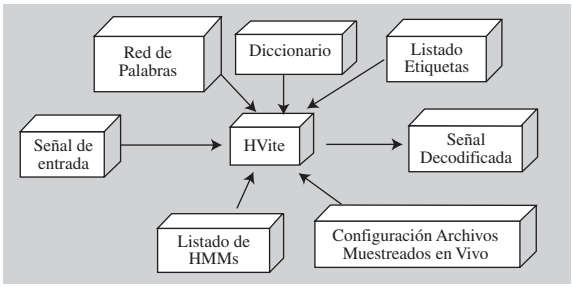

Figura 14. Diagrama en bloque para llamar a HVite.

HResults. Es la herramienta de HTK, que entrega los resultados del proceso de evaluación. Compara las etiquetas de entrada y las de salida; dando lugar a una matriz de confusión de los distintos HMM (calidad del proceso). La salida de este comando es realizada por pantalla y para guardar los resultados habrá que usar un

redireccionamiento adecuado hacia un archivo de texto. Ver figura 15.

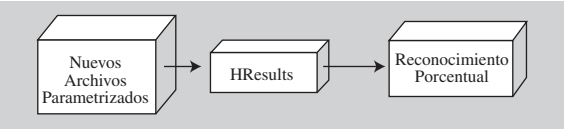

Figura 15. Diagrama en bloque para llamar a HResults.

EPÍLOGO

Se ha dado a conocer la herramienta de software HTK, la cual, por ejemplo, podemos aplicar para resolver problemas en el área del reconocimiento automático del habla, o en otras, donde la problemática pueda enfocarse desde una perspectiva estocástica.

En un próximo trabajo se darán a conocer los resultados obtenidos en utilizar HTK, para el reconocimiento silábico en nuestro lenguaje.

CONCLUSIONES

HTK es una poderosa herramienta para la creación y manipulación de HMMs. Es ideal para experimentar con modelos de lenguajes y modelos de Markov. Permite, además, probar diferentes metodologías de entrenamientos. Otra gran ventaja es su gran capacidad como sistema etiquetador, logrando resultados muy eficientes. Con él se puede etiquetar cualquier base de datos lingüística (presentes y/o futuras). Esto permitiría entrenar sistemas más robustos.

Respecto de sus desventajas, debo mencionar en primer lugar: la poca amabilidad del sistema para realizar de manera más sencilla las llamadas a librerías y paso de parámetros que requiere el sistema. El manejo de archivos resulta bastante hostil para los usuarios neófitos al ambiente Unix/Linux. Pero no tanto para usuarios DOS, activado desde una plataforma Windows.

AGRADECIMIENTOS

El autor desea agradecer de manera especial al CMCC (Centro de Modelación Científica y Computacional), de la Universidad de La Frontera, por todo el apoyo brindado en la ejecución de este proyecto.

REFERENCIAS

[1] S. Young, D. Kershaw, J. Odell. “The HTK Book”. V3.2. CUED. UK. July 2004.

[2] B. Resch. “Automatic Speech Recognition with HTK”. Signal Processing and Speech Communication Laboratory. Inffeldgase. Austria. Disponible en Internet: http://www.igi.tugraz.at/lehre/CI

[3] L. Rabiner, B.H Juang. “Fundamentals of Speech Recognition”. Prentice Hall. NY, USA. 1993.

[4] R. Barrientos, C. Zamora. “Reconocimiento de Palabras Aisladas, Usando Modelos Ocultos de Markov”. Tesis para optar al título de Ingeniero Civil Electrónico. Universidad de La Frontera. Temuco, Chile. 2004.

[5] J. Proakis, Ch. D. G. Manolakis; “Tratamiento Digital de Señales”. Prentice - Hall. 1998.

[6] A. Oppenheim, R. Schafer. “Discrete-Time Signal Processing”. Prentice-Hall. USA. 1989.

[7] A. Procházka, J. Uhlír and P. Sovka. “Signal Analysis and Prediction I”. Procházka et al. Prague, Czech Republic. 1998.

[8] M. Karnjanadecha, S. Zahorian. “Signal modeling for High-Performance Robus Isolated. Word Recognitions”. IEEE Transactions On speech and Audio Processing. Vol 9 Nº 6. September 2001.