Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Keyword spotting en idiomas sin datos de entrenamiento

Tesis presentada para obtener el título de
Licenciado en Ciencias de la Computación

Pablo Brusco

Directores: Luciana Ferrer y Agustín Gravano

Buenos Aires, Mayo 2014

Keyword spotting en idiomas sin datos de entrenamiento

En este trabajo se propone estudiar el problema llamado detección de palabras claves (keyword-spotting en inglés) en el contexto de idiomas que no disponen de corpus de datos con grabaciones y transcripciones temporales o fonéticas. El desarrollo y experimentación han sido realizados utilizando el Boston University Radio Speech Corpus, una base de datos de grabaciones tomadas de una radio en Massachusetts. Se analiza el uso de modelos ocultos de Markov (HMMs) para la detección de palabras sobre habla continua estudiando diversas topologías y parametrizaciones. Los modelos se basan en el uso de "fillers" para palabras no buscadas y palabras completas o fonemas como unidades mínimas de detección. Los resultados muestran que el mejor modelo alcanza rendimientos superiores a un 0.47 de FOM promedio, un porcentaje de detecciones correctas del 72.1 % y 78.9 falsas alarmas por hora. Para las pruebas, se utiliza un conjunto de 20 keywords entrenadas con 14 minutos de datos transcriptos y fillers entrenados con 7 horas sin transcripciones. Los resultados se muestran en base a 1.9 horas de datos para testeo.

Palabras claves: Detección de Palabras Claves, Keyword-Spotting, Speech-Recognition, Fillers, Modelos Ocultos de Markov, Word-Spotting

AGRADECIMIENTOS

- A mis directores: Agustín y Luciana por la infinita paciencia y dedicación que hicieron que fuera un gusto hacer el trabajo.
- Gracias para "el Agus", "la Naty", "la Sonia", "el Quique" y el resto de mi familia de cordobeses que bancaron mis días como zombie y me brindaron muchísimo apoyo.
- A "la Elvi" por toda su compañía, paciencia y tanto amor durante estos años.
- A mis amigos de la vida: Gusti, Mati, Mati (Légolas), Nico, Santi que me acompañaron y ayudaron en todo momento.
- A Mariano Bianchi, mi fiel compañero de cursadas, estudios, ping-pones y comidas al disco desde las profundidades de Luján. Claramente el responsable de los muchos empujones que necesitaba para terminar la carrera.
- Al equipo de Manas que me proporcionó miles de herramientas útiles durante cuatro años que me sirvieron para poder avanzar en la carrera y en la vida profesional.
- A Ariel y Miriam, responsables de que haya elegido esta carrera que tanto adoro.
- A mis alumnos y compañeros docentes que hacen que nunca me canse de ir a las clases y me hacen sentir que el trabajo no es trabajo.

Índice general

1. INTRODUCCIÓN

1.1. Definición del problema

Existe una ONG llamada Farm Radio International que se encarga de recolectar y fomentar programas radiales africanos en distintos idiomas que hablen de cómo cultivar, cosechar, trabajar en granjas/campos, etc. En esta organización enfrentan un problema, el contenido que recolectan no está indexado ni clasificado por temas, y si alguien necesita identificar un programa en el que se habló de un aspecto en particular, no tienen manera de buscar en este contenido más que escuchando programas enteros.

De aquí surge la idea de poder ofrecer a un oyente la posibilidad de escuchar grabaciones de radio que contengan un tema en particular. Por ejemplo, para el caso de las plagas de langostas en campos de tomate, en aquellos programas que contengan las palabras claves (indistintamente llamadas *keywords* de aquí en adelante) "langostas", "tomate", "plaga", se podrá escuchar momentos relevantes en los cuales se mencionan estas palabras.

Por lo tanto, se necesitaría desarrollar un sistema que, dado un conjunto de keywords; un corpus de programas de radio grabados directo del estudio radial sin transcripciones asociadas; y entrenadores –personas que estén dispuestas a grabar o marcar en audio existente cada keyword las veces que el sistema necesite–, devuelva las ocurrencias de estas keywords en los programas para que puedan ser indexados y ofrecidos al oyente por teléfono.

Este sistema no permitirá búsquedas de keywords dichas por el usuario directamente, sino que se les ofrecerá una serie de temas posibles que estarán previamente asociados a ciertas keywords ya disponibles gracias al trabajo de los entrenadores.

Una vez entrenado, si se agregaran nuevo programas, el sistema debería poder analizarlos sin requerir nuevas grabaciones de los entrenadores. En caso de agregar una keyword, el sistema podría requerir las grabaciones de los entrenadores junto a un nuevo proceso de entrenamiento y luego detectar estas nuevas keywords en los programas.

Adicionalmente, se establecerá la restricción principal del sistema que consiste en evitar utilizar grandes cuerpos de datos con transcripciones o modelos preexistentes del habla ya que se asume que el idioma con el que se trabajará no posee este tipo de recursos (como sí poseen lenguajes como el inglés o el español y muchos otros). Por lo tanto, más allá de utilizar el idioma inglés en esta tesis por poseer un corpus apropiado para el estudio, se tendrá en mente un idioma como el suajili, idioma principal del Este de África.

En base a las especificaciones anteriores, este trabajo se encargará de analizar la factibilidad de diseñar este sistema utilizando una de las técnicas de modelado más populares, hasta el momento para este tipo de herramientas, es decir, Modelos Ocultos de Markov.

En resumen, el problema que se desea resolver es el de keyword-spotting (también llamado word-spotting) que consiste en detectar palabras claves en grabaciones de habla continua donde, en la mayoría de los casos, estas palabras son grabadas para su búsqueda u obtenidas de otras grabaciones. Este problema es conocido e incluso anterior al de Reconocimiento Automático del Habla (o ASR por sus siglas en inglés).

El ASR consiste en transcribir palabras habladas a texto, y constituye un problema muy estudiado en la computación. Sus algoritmos son utilizados diariamente en todo tipo de campos como, por ejemplo, reconocimiento de comandos en teléfonos celulares y en miles

de aplicaciones comerciales. El keyword-spotting puede ser considerado un sub-problema del ASR y, por lo tanto, utilizando los mismos algoritmos podemos transcribir a palabras todo el audio donde deseamos ubicar las keywords y luego buscar las ocurrencias de estas en las transcripciones.

De todas formas, este último mecanismo tiene una gran desventaja, el reconocimiento general de habla no es un problema fácil de resolver y solo funciona bien bajo una gran cantidad de supuestos. Entre estos supuestos se encuentra el de utilizar grandes cuerpos de datos de entrenamiento (es decir, habla transcripta a nivel de palabras e incluso, para mejores resultados, a nivel fonemas) que suelen escasear en la mayoría de los lenguajes, en particular lenguajes como el suajili.

En función de lo anterior, el objetivo de este trabajo consiste en estudiar posibles alternativas para abordar este problema minimizando los recursos disponibles, tratando de emular las condiciones reales que podrían surgir en caso de intentar volver realidad el sistema antes detallado.

1.2. Trabajo Previo

Tanto en reconocimiento de palabras claves como en reconocimiento general del habla, trabajos como el de Bridle (1973) se basaron en el uso de *Dynamic Time Warping* (DTW). DTW es una técnica que utiliza programación dinámica para medir similitudes entre dos secuencias temporales que pueden tener desfasajes, en este caso, dos secuencias de audio en búsqueda de coincidencias. Esta técnica probó ser útil cuando el reconocimiento se hace sobre datos parecidos a los de entrenamiento, en cuanto se agrega variabilidad, demostró no escalar. Otro aporte importante presentada por Bridle fue la utilización de modelos de *filler*, representación de palabras fuera del conjunto de palabras buscadas.

Más recientemente, a partir del trabajo de Rabiner (1989), se ha popularizado el uso de Modelos Ocultos de Markov de distribuciones continuas utilizando palabras o sub-palabras como unidades acústicas. Desde entonces ha sido la técnica más utilizada para reconocimiento del habla. El uso de sub-palabras facilita el reuso de entrenamiento entre palabras que comparten ciertas partes permitiendo, por ejemplo, el reconocimiento de palabras fuera del conjunto de entrenamiento.

En el caso del reconocimiento de palabras claves, el uso de estos modelos fue aplicado por primera vez por Rohlicek y col. (1989) y Rose y Paul (1990) marcando el comienzo de numerosos estudios basados en sus ideas. Cabe destacar que el trabajo de Rose y Paul (1990) indica que para conseguir una buena performance se necesita entrenar un modelo de filler complejo utilizando grandes cantidades de datos transcriptos o modelos preexistentes, en particular utilizaron un HMM que conectaba completamente todo los posibles modelos de sub-unidades del lenguaje. Por otro lado, hay trabajos como el de Weintraub (1993) que complejizan aún más estos modelos utilizando modelos de vocabulario completo que excluyen a las keywords.

Esta técnica funciona muy bien pero requiere una gran cantidad de datos de entrenamiento. Como se dijo en la sección anterior, el objetivo de esta tesis es evaluar un sistema pensado para utilizar la menor cantidad de datos transcriptos posibles. Trabajos como el de Moore (2003) muestran que un sistema de reconocimiento general disminuye hasta un 15 % de efectividad al trabajar con poca cantidad de datos (usando bases de datos de menos de 100 horas).

Artículos como el de Das e India (2005) exponen maneras de construir sistemas de

detección de palabras claves utilizando la herramienta HTK. También muestra alternativas para representar fillers y modela keywords utilizando palabras como unidad acústica con Modelos Ocultos de Markov de 15 estados.

El problema de detección de palabras claves tiene ciertas características que hacen posible realizar un sistema con corpus mucho menores a los necesarios para el reconocimiento general de habla, manteniendo una performance razonable. Existen trabajos basados en la idea de utilizar la menor cantidad de datos. Por ejemplo Thambiratnam (2005) dedica un capítulo de sus tesis de doctorado en evaluar sistemas para lenguajes distintos al inglés. En particular, como se ve afectado el sistema al variar entre 164, 15 y 4 horas de entrenamiento en inglés, español e indonés. Además, compara distintos modelos para representar fonemas en las keywords, uno que utiliza *trifonos* y otros que utilizan *monofonos*, explicando que solo esta última alternativa es viable para una baja cantidad de datos de entrenamiento. Su trabajo concluye que el reconocimiento de palabras claves sufre menos en performance ante la falta de datos que las tareas de reconocimiento general del habla.

También, el artículo de Garcia y Gish (2006) explica que con solo 15 minutos de transcripciones junto a un modelo de reconocimiento que define sus propias unidades de sonido en base a modelos segmentales se puede alcanzar una gran performance. De todas maneras, en esta tesis decidimos utilizar la manera estándar de Modelos Ocultos de Markov con distribuciones continuas que han probado funcionar muy bien con grandes cantidades de datos.

Como resultado, esta tesis se enfocará en atacar el problema utilizando Modelos Ocultos de Markov continuos para representar keywords y fillers con un esquema de filler simple y keywords entrenadas en base a apariciones en menos de 14 minutos de grabaciones transcriptas utilizando monofonos para representar los fonemas presentes en las keywords.

1.3. Estructura

El capítulo 2 introduce las técnicas utilizadas para resolver el problema de detección de palabras claves. En este capítulo se detalla la base teórica necesaria para comprender la solución planteada.

En el capítulo 3, se detalla el corpus de datos utilizado para los experimentos, luego se introduce el funcionamiento del sistema junto con las métricas utilizadas en la experimentación.

En el capítulo 4 se muestran los resultados obtenidos junto con una discusión final de estos.

La tesis concluye con el capítulo 5 en el cual se realiza un resumen y se presenta el posible trabajo futuro.

2. TÉCNICAS UTILIZADAS

El reconocimiento automático del habla es un área de estudio dentro de la Inteligencia Artificial que busca conseguir que las computadoras puedan utilizar información obtenida de sonidos que contienen habla humana y poder procesarla para objetivos variados como dictado automático, reconocimiento de comandos, traducción automática, reconocimiento de palabras claves, etc.

Generalmente, el audio está almacenado en formato digital y se necesita procesarlo para conseguir extraer las características necesarias para la tarea.

Una vez que se pudo extraer la información principal del audio, dada la enorme variabilidad del habla humana, se utilizan modelos estadísticos que permiten a los programas de reconocimiento tomar decisiones sobre el contenido y poder, así, cumplir con el objetivo del sistema. Para el caso de este trabajo, los modelos utilizados se conocen como Modelos Ocultos de Markov (HMMs por sus siglas en inglés).

En general, la dificultad de la tarea de reconocimiento del habla está ligada al tipo de sistema que se desea realizar. Sistemas como reconocedores de dígitos dependientes del hablante son relativamente sencillos, en cambio, sistemas de reconocimiento sobre habla continua con vocabulario amplio, independiente del hablante, son considerados de gran dificultad (Jurafsky y Martin 2009).

A continuación se detalla este proceso, haciendo hincapié en los métodos que aplican al reconocimiento de palabras claves. El contenido de este capítulo se basa en los capítulos 6 y 9 del libro Speech and Language processing de Jurafsky y Martin (2009).

2.1. Arquitectura general

Para el reconocimiento del habla basada en HMMs suele utilizarse la metáfora del *canal ruidoso*; la intuición es tratar a la onda de audio como una versión *ruidosa* de una cadena de palabras, es decir, una versión que atravesó un canal que distorsionó el texto haciendo dificultoso reconocer el original.

La tarea, entonces, consiste en construir un modelo del canal que permita determinar cómo éste modificó la cadena y, por lo tanto, poder recuperarla. La base consiste en que si conociéramos cómo este canal distorsiona la fuente, podríamos determinar el texto correcto tomando cada posible oración del lenguaje, pasándola por el modelo del canal y comprobando qué tan bien se corresponde contra la salida. Finalmente, seleccionamos la fuente que tuvo mejor coincidencia, como nuestra cadena "correcta".

Para implementar este modelo necesitamos solucionar una serie de problemas. Primero, existe una dificultad técnica a resolver que consiste en obtener la información esencial de las grabaciones para luego poder entrenar y comparar modelos, a este proceso se lo conoce como *Extracción de características principales.*

En segundo lugar, la cantidad de oraciones posibles en un lenguaje es inmanejable, por lo cual necesitamos algoritmos eficientes que no busquen sobre todas las posibilidades, sino que sólo consideren las que tienen buenas chances de coincidir con la entrada. A este problema se lo conoce como *Problema de la Decodificación.*

Por otra parte, para elegir la oración que mejor coincida con la entrada, necesitaremos una métrica. Como se mencionó anteriormente, dada la variabilidad del habla, se

utilizará un modelo estadístico para realizar estas comparaciones. Lo anterior implica que el problema del reconocimiento del habla es un caso especial de inferencia Bayesiana y requiere combinar varios modelos probabilísticos para obtener una estimación completa de la probabilidad de una observación, dada una secuencia de candidatos posibles.

En las siguientes secciones se detallará cada uno de estos problemas que presentan la base teórica que se aplica luego en la experimentación.

2.2. Extracción de características principales

Para procesar audio y obtener información del habla se utilizan varias técnicas conocidas. En el caso de reconocimiento continuo del habla, la más utilizada es la extracción de *Mel Frequency Cepstral Coefficients* (MFCCs) que consta de seis pasos básicos que transforman la señal digital en un vector de características principales (*feature vector*) que contiene una muestra de la señal, permitiendo así conservar ciertos rasgos relevantes para luego extraer información del audio. Por ejemplo, si lo que se busca es detectar palabras, se intenta eliminar ciertas propiedades del audio como el tono y timbre de voz que no aportan a la tarea; aunque si lo que se busca es detectar quién es el hablante, remover este dato puede resultar contraproducente. Los pasos principales son:

1. Preemphasis
2. Windowing
3. Transformada discreta de Fourier
4. Mel Filter Bank y logaritmo.
5. Inversa de la transformada discreta de Fourier
6. Deltas y Energía

El primer paso utiliza un filtro que permite estabilizar la energía que se produce en distintas frecuencias, dado que el pulso glotal otorga mayor energía a las frecuencias bajas, empeorando los modelos posteriores. El segundo proceso, Windowing, permite extraer porciones de la señal donde se asume que las propiedades estadísticas son constantes en el tiempo recorriendo el audio y tomando muestras de, en general, 25 milisegundos cada 10 milisegundos. Luego, para evitar que la transformada de Fourier de la señal sobre la ventana tenga altas frecuencias introducidas por la discontinuidad en los bordes, se utilizan ventanas especiales llamadas ventanas de Hamming que evitan la discontinuidad.

En tercer lugar, se aplica la transformada discreta de Fourier para obtener la información espectral de la señal. Esta se aplica sobre las ventanas resultantes del proceso anterior y finalmente se obtiene, para cada banda de frecuencia, un número complejo que representa la magnitud y fase de esa frecuencia en la señal original. En general se utiliza el algoritmo FFT (fast Fourier transform). Una vez que se dispone de la cantidad de energía para cada banda, se mapea a una escala llamada *mel* que sirve para hacer equidistantes sonidos que para el oído humano suenan de esa manera, mejorando así los modelos posteriores.

Finalmente, se aplica la inversa de la transformada discreta de Fourier para terminar de eliminar características no esenciales para esta tarea de reconocimiento del habla. Luego, se extraen 12 componentes de este resultado que representan la información necesaria

del tracto vocal, limpia de información provista por la fuente glotal. Una característica importante de estos coeficientes es que no están correlacionados entre sí permitiendo, más adelante, utilizar modelos más simples reduciendo gran cantidad de parámetros (en particular en las mezclas de Gaussianas). Una vez conseguidas las características principales, se calcula la energía total y se toma la derivada primera y segunda de cada componente (y energía) para lograr capturar velocidad y aceleración de cambio entre frames cercanos. Por lo tanto, se trabajará de aquí en más con vectores de 39 coordenadas que representarán la información necesaria para la detección.

2.3. Modelo Estadístico

Siguiendo la idea de la metáfora del canal ruidoso, se quiere descubrir el conjunto de palabras más probable $\hat{W} = \hat{w_1} \dots \hat{w_n}$ con $w_i \in \mathscr{L}$[1] que puede haber generado un conjunto de observaciones acústicas $O = o_1 \dots o_t$. Por lo tanto, lo que se quiere obtener es $\hat{W}$ tal que maximice la probabilidad $P(W|O)$. Formalmente,

$$\hat{W} = \underset{W \in \mathscr{L}^*}{\operatorname{argmax}} P(W|O)$$

utilizando Bayes, podemos convertir esta expresión en

$$\hat{W} = \underset{W \in \mathscr{L}^*}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)}$$

además, $P(O)$ no depende de W, por lo tanto maximizar con o sin el dividendo es equivalente, es decir

$$\hat{W} = \underset{W \in \mathscr{L}^*}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)} = \underset{W \in \mathscr{L}^*}{\operatorname{argmax}} P(O|W)P(W)$$

Quedan así dos componentes, $P(W)$ que representa la probabilidad de una frase en nuestro lenguaje y $P(O|W)$ que indica la probabilidad de que una grabación provenga de una oración dada. Veremos cómo podemos calcular cada una y cómo, luego, calcular el argumento que maximiza la cuenta anterior.

Para el cálculo de $P(W)$ suelen utilizarse diversas técnicas dependiendo de distintos niveles de complejidad. En general, reconocedores de habla continua suelen utilizar modelos de *Trigramas* con la gran desventaja de requerir muchos datos para el entrenamiento. Ya que no contamos con ese privilegio por apuntar a lenguajes como el suajili, el modelo que utilizaremos es una simple gramática con probabilidades uniformes entre sus componentes.

Por otra parte, para conocer $P(O|W)$ se construyen modelos acústicos que permiten computar la probabilidad de una observación dada una unidad lingüística como por ejemplo, una palabra, un fonema, una sílaba, etc.

2.3.1. Modelo Acústico

El modelo acústico que utilizaremos en este trabajo tiene como nombre *Modelo Oculto de Markov* (HMM por sus siglas en inglés), y se trata de un autómata finito compuesto por

[1] $\mathscr{L}$ representa el conjunto de palabras del lenguaje

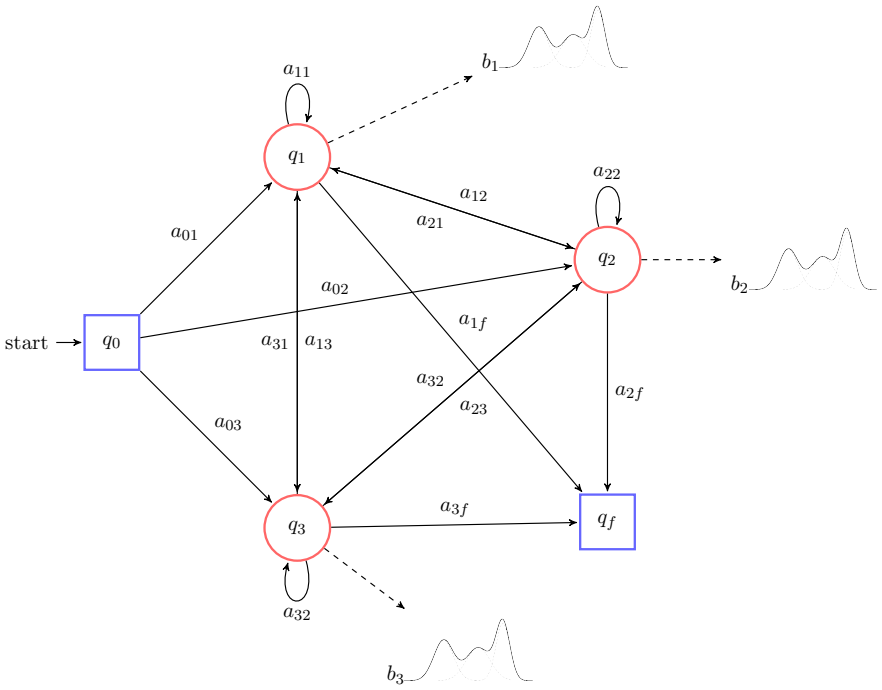


Fig. 2.1: Estructura de un Modelo Oculto de Markov

- $Q = q_1 \dots q_n$ un conjunto de estados;
- $A = a_{01}a_{02} \dots a_{1n} \dots a_{nn}$ una matriz de probabilidades de transición entre estados. Cada a_{jk} representa la probabilidad de transición desde el estado j y hacia el estado k;
- $O = o_1 \dots o_m$ un conjunto de observaciones;
- $B = b_i(o_t)$ un conjunto de probabilidades de emisión, representa la probabilidad de que el estado i emita la observación t, es decir $P(o_t|q_i)$;
- q_0, q_f estados especiales de inicio y fin sin observaciones asociadas.

En la figura 2.1 puede verse una representación general en donde se pueden observar tres estados emisores junto con el resto de las componentes de estos modelos.

Los HMMs se basan en 2 presunciones fuertes:

1. Suposición de Markov: $P(q_i|q_1 \dots q_{i-1}) = P(q_i|q_{i-1})$
2. Independencia de emisiones: $P(o_i|q_1 \dots q_i \dots q_n, o_1 \dots o_i \dots o_t) = P(o_i|q_i)$

Estas presunciones encajan bien con nuestro problema del habla, dadas las características que posee. Veamos un ejemplo para ver cómo esta representación permite solucionar nuestro problema.

Supongamos que contamos con un HMM especializado en reconocer la palabra “apple”. Todavía no hemos dicho cómo se construye, pero pensemos que se trata de un HMM entrenado; es decir, A (probabilidades de transición) y B (probabilidades de emisión) están definidos.

Utilizaremos la figura 2.2 para representar a este HMM que se basa en la composición fonética de `APPLE`: `ae p ah l`. Por ahora, cada estado representará a un fonema, luego veremos otra alternativa.

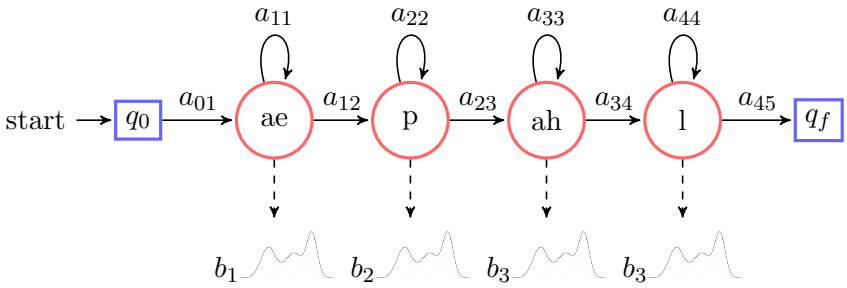


Fig. 2.2: HMM para reconocer "apple"

Para comenzar el reconocimiento de la palabra, la onda sonora será convertida a una gran secuencia de vectores MFCCs, como se explicó en la sección 2.2, que será comparada contra este HMM para obtener la probabilidad de que estos sonidos hayan sido emitidos por este autómata.

Para ello, en la etapa de entrenamiento, se desarrollarán distribuciones probabilísticas para cada estado, $b_i(o_t) = P(o_t|q_i)$ donde o_t es un vector MFCC y q_i un estado, las cuales deberán estimar la probabilidad de que un vector de 39 componentes, proveniente de 10 milisegundos de habla humana, haya sido emitido por ese estado.

Por este motivo, se utiliza un tipo de distribución probabilística que ha probado ser efectiva para este tipo de tareas y, por lo tanto, una de las técnicas más utilizadas: *Modelos de Mezclas Gaussianas* que consta de sumar varias distribuciones normales para ajustarse a la variabilidad de los vectores acústicos que no siguen una distribución normal.

Mientras más Gaussianas se utilicen para componer una mezcla, más flexible será al momento de reconocer sonidos. Por otro lado, aumentar esta cantidad implica un mayor costo computacional y una mayor cantidad de entrenamiento. Veremos en el capítulo 3 cómo variando esta cantidad de mezclas lograremos modelar keywords y fillers lo suficientemente flexibles para la detección de palabras claves.

En resumen, cada estado contará con una distribución asociada que servirá para detectar dichos fonemas.

2.3.2. Topología de los modelos y fillers

Al revisar un espectro de voz se puede ver que los fonemas no son uniformes a lo largo del tiempo. Si se los examina, se pueden ver tres áreas: la transición del fonema anterior al actual, la parte central y la transición al siguiente fonema. Por esta razón suele utilizarse tres estados por fonema (inicio, medio y final), consiguiendo así mejor precisión para el reconocimiento. La figura 2.3 muestra la modificación necesaria al esquema anterior para representar este cambio.

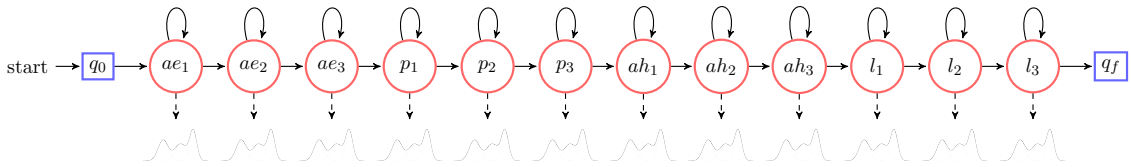


Fig. 2.3: HMM para reconocer "apple" con fonemas divididos

Puede notarse, en estos ejemplos, que cada estado contiene una transición hacia el siguiente estado y, además, un ciclo.

Constrastando con el ejemplo de la figura 2.1, no todas las transiciones han sido dibujadas, esto se debe a la topología generalmente elegida para las palabras claves, topología *left-right* (llamada *Bakis Network*). A diferencia de la topología *ergódica* que estudiaremos para representar fillers, esta topología no permite retroceder ni saltearse estados, modelando así la naturaleza secuencial del habla.

Los ciclos permiten que un sub-fonema se repita indefinidamente, capturando así la variabilidad en el tiempo de la entrada acústica. Según (Jurafsky y Martin 2009), el sonido de fonemas como [aa] varia de 7 a 387 milisegundos, lo cual equivale a un rango entre 1 y 38 vectores MFCC.

Hasta aquí hemos mostrado cómo un HMM puede modelar palabras claves, veamos ahora otro tipo de modelo, los fillers.

El uso de fillers permitirá capturar cualquier sonido presente en las grabaciones, es decir, deberá ser lo más general posible. Para lograrlo, suelen utilizarse diversas técnicas como construir un solo HMM entrenado con una gran cantidad de ejemplos, también se usa la alternativa de tener múltiples fillers representando a cada fonema del lenguaje o múltiples fillers para los distintos alófonos del lenguaje, etc[2].

En este trabajo se utiliza un HMM ergódico, es decir, un modelo donde todos los estados que emiten estarán conectados entre sí a diferencia de la topología anterior. Si quisiera hacerse un esquema, sería similar al de la figura 2.1. En ese caso, estaríamos en presencia de un modelo de filler de cinco estados (tres emisores).

2.3.3. Entrenamiento y Decodificación

Una vez que está clara la estructura que se utilizará para cada modelo, se procede a la etapa de entrenamiento, es decir, la estimación de los parámetros del HMM. La manera de entrenar el modelo será mediante el algoritmo de Baum-Welch llamado *forward-backward*, un algoritmo iterativo que intenta encontrar los parámetros que maximicen la probabilidad de haber generado los datos de entrenamiento. Este algoritmo estima las probabilidades asociadas a cada estado y las probabilidades de transición entre estados.

Como entrada, recibe grabaciones etiquetadas a nivel fonemas que le permitirán asociar sonidos con los estados correspondientes y sus transiciones. En el caso de las keywords, suponiendo que no dispondremos de transcripciones a nivel fonema, pero sí a nivel palabras, se divide cada palabra uniformemente según diccionarios fonéticos.

Este último paso es posible en lenguajes en los que se dispone de dicho diccionario. Para los casos en que no se disponga, habrá que hacer un estudio del lenguaje para ver

[2] ver el trabajo de Das e India (2005) por ejemplo.

si es factible utilizar la traducción ortográfica y luego mapear a los fonemas correspondientes. En el caso del español y el suajili es razonable efectuar dicha operación por las características de estos lenguajes, en los cuales existe un mapeo casi inmediato de letras a sonidos.

Finalmente, cuando nuestros modelos están entrenados se los conecta utilizando el modelo de lenguaje (como habíamos mencionado anteriormente, mediante una gramática con probabilidades uniformes) quedando listo para la tarea de reconocimiento.

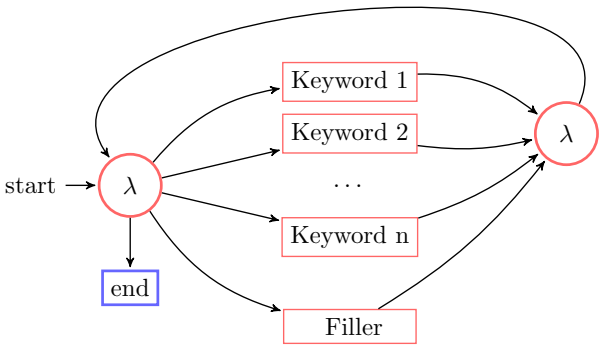


Fig. 2.4: Gramática de estados finitos

La gramática de estados finitos mostrada en la figura 2.4 refleja los posibles estados en los que puede estar el proceso de reconocimiento sobre una grabación. Este tipo de gramáticas es una de las alternativas existentes para modelar problemas de reconocimiento de keywords. En este caso en particular, se permite una cantidad indeterminada de repeticiones de keywords en donde puede o no haber apariciones de fillers intercaladas.

La figura 2.5 muestra una ampliación de la gramática utilizando dos keywords ("one" y "five") junto a los HMMs vistos sobre el cual el algoritmo actúa[3].

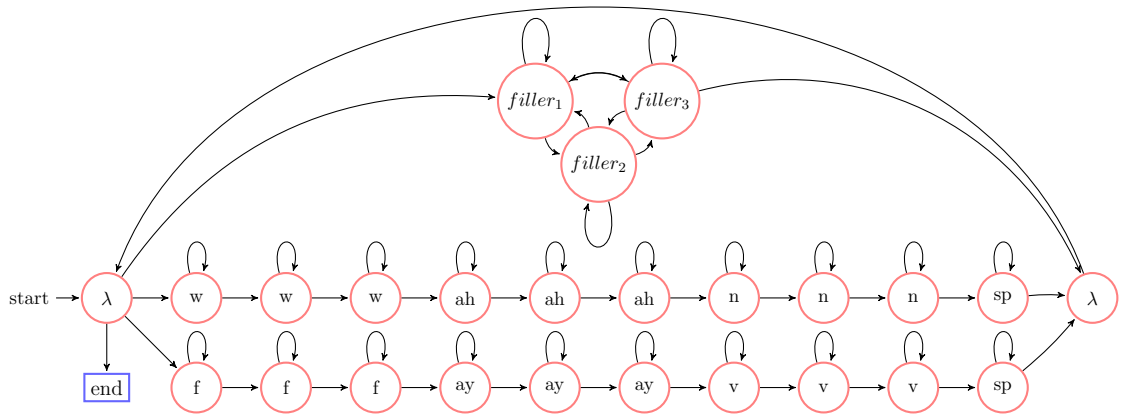


Fig. 2.5: HMMs unificadas

[3] En este esquema no se dibujaron los estados no emisores para simplificar la explicación

El algoritmo de *Viterbi* (decodificación), recorre el HMM buscando el mejor camino respecto de un conjunto de datos otorgados que seguirán siendo MFCCs pero esta vez sin transcripciones asociadas. De esta manera, devuelve las palabras que se forman utilizando el camino más probable junto con un puntaje que utilizaremos más adelante.

2.4. Unidad acústica

Al momento de decidir la representación para el modelo de keywords, surgen varias alternativas utilizadas comúnmente en el reconocimiento del habla. Hasta ahora veníamos suponiendo que cada keyword se descompondría en fonemas y luego modelaríamos cada fonema utilizando tres estados internos (*Subword Acoustic Model*).

Otra alternativa es la de utilizar palabras como unidades mínimas (*Word Acoustic Model*). En este caso, cada HMM modela y es entrenado con palabras completas, con la cantidad de estados necesarios. De esta manera, el sistema "aprende" la representación completa de la palabra.

La alternativa de utilizar palabras como unidad tiene dos ventajas claras. Una es no necesitar de un diccionario fonético, no disponible en muchas ocasiones. La otra ventaja consiste en que, dado que ciertos fonemas tienen distinta forma según el contexto en donde se producen, es más fácil el reconocimiento cuando no se involucran estas variaciones.

También se conocen varias desventajas esenciales de esta alternativa. Primero, requiere gran cantidad de datos de entrenamiento, ya que lo que se quiere aprender tiene mayor longitud y se necesitan repeticiones lo suficientemente variadas como para otorgar confianza de que se reconocerá la palabra en cualquier contexto o al menos en varias situaciones distintas. Por otra parte, muchas palabras comparten sus fonemas, por lo tanto, es un desperdicio no utilizar la redundancia en estos datos que, además, permitirían reconocer nuevas palabras sin necesidad de volver a entrenar el sistema.

Es esta tesis utilizamos mayormente la alternativa de división por fonemas salvo en algunas pruebas en donde se especificará lo contrario.

3. EXPERIMENTOS

A continuación se describe el corpus de datos, y se explica el sistema implementado.

3.1. Corpus de datos

El corpus de datos utilizado es el Boston University Radio Speech Corpus (Ostendorf, Price y Shattuck-Hufnagel 1995). Consiste en grabaciones radiales de lecturas de noticias hechas por locutores profesionales. Consta de siete lectores (cuatro hombres y tres mujeres) dedicados a anunciar noticias en radio FM, que produjeron grabaciones en dos etapas: por un lado, en un laboratorio de la Universidad de Boston donde se les pidió que leyeran 24 historias alternando su manera de hablar entre forma radial y no radial; por otro lado, más de siete horas de audio tomadas directamente del programa radial en los estudios de la WBUR (una radio local de Boston). El audio fue digitalizado a 16kHz utilizando un conversor analógico digital de 16 bits.

Tanto las historias leídas como los programas de radio fueron anotados con transcripciones ortográficas generadas a mano con precisión a nivel palabra. Es decir, cada palabra está marcada con un tiempo que indica el momento en que terminó de decirse la misma en la grabación. Luego, se estima el comienzo de la palabra siguiente tomando el fin de la inmediata anterior.

3.2. Desarrollo del sistema

Para realizar los experimentos se desarrolló un sistema basado en la popular herramienta Hidden Markov Model Toolkit (HTK), paquete que integra funciones útiles para trabajar con Modelos Ocultos de Markov (Young y col. 2006).

Sobre esta herramienta se diseñaron una serie de scripts que luego fueron migrando a convertirse en una aplicación en el lenguaje Ruby que facilitó el uso de HTK encapsulando las funcionalidades útiles con respecto al trabajo que se deseaba realizar. También brindó un marco en el cual era posible realizar tests de unidad e integración para aumentar la confianza en el sistema.

3.2.1. Definición de sets de datos

El primer paso al plantear el sistema consistió en determinar qué parte del corpus sería tomada para entrenar los modelos, qué parte para testear y cuál para determinar los resultados finales.

Analizando el tamaño y la escasa cantidad de locutores, se tomó la decisión de dividir al corpus en seis grupos uniformes. Para ello, se listaron todos los archivos de grabaciones ordenados alfabéticamente y se recorrieron asignando cada uno a un grupo de pertenencia, entre 1 y 6. De esta manera, la cantidad de grabaciones por grupo pertenecientes a cada hablante y en cada situación es pareja. Una vez separados, se tomaron 4/6 del total (7.31 horas) de las frases para entrenamiento, 1/6 (1.8 horas) para testeo y 1/6 (1.9 horas) para resultados finales que se ignoran durante todo el desarrollo del sistema. Junto con esta

división, se adjuntaron a cada grabación las transcripciones correspondientes que permiten realizar, tanto el entrenamiento, como la recolección de resultados de estos.

También se realizó una división del corpus por hablante para una de las pruebas finales, en este caso se dividió el corpus por hablante y se hizo una rotación en la cual se selecciona de a un hablante y luego se toman todos sus archivos para testeo y el resto del corpus para entrenamiento. De esta manera se hace un estudio del sistema cuando el hablante no es parte del corpus de entrenamiento.

3.2.2. Preparación del entrenamiento

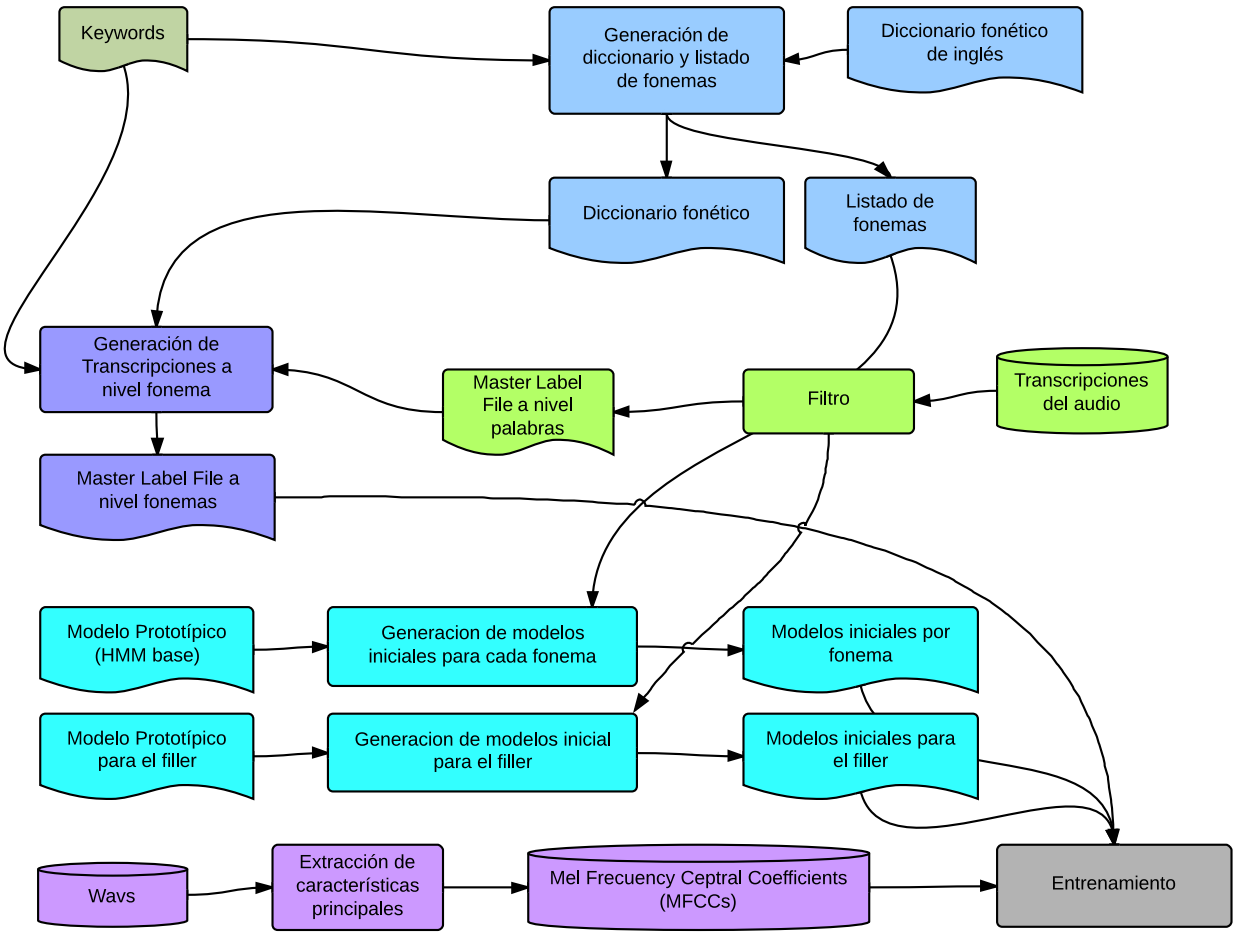


Fig. 3.1: Preparación para el entrenamiento

Para lograr la generación de HMMs que modelen los fonemas o palabras que debemos reconocer, se necesitan una serie de pasos que van desde la elección de las keywords que el sistema reconocerá hasta la creación de modelos prototípicos que permitirán comenzar el entrenamiento. La figura 3.1 muestra un panorama general de las tareas que se realizan. A continuación se detalla el proceso.

El primer paso consiste en seleccionar las palabras que se utilizarán como keywords

y luego, usando la herramienta HDMan de HTK y tomando un diccionario fonético del idioma a entrenar como base, se crean dos archivos primordiales para continuar; el diccionario que contiene las traducciones de palabras clave a sus fonemas y el listado completo de fonemas utilizados.

Además, se cuenta con las transcripciones de las grabaciones que deben ser procesadas para lograr obtener un Master Label File (MLF). Un MLF contiene toda la información relacionada con las transcripciones asociadas a cada grabación y es, en general, la concatenación de las transcripciones individuales separadas por nombres de archivos. Pero en el caso de este sistema, este mecanismo no representa la realidad que desea emularse ya que los datos con los que se contará para entrenar modelos reales serán repeticiones de keywords y no audio completamente transcripto. Por lo tanto, se filtran del MLF todas las palabras que no sean keywords dejando espacio que será llenado con fillers.

La manera de introducir los fillers en el sistema es precisamente completando estos huecos que dejó el filtro de keywords. Huecos que se sabe contienen habla pero no se procesa como audio para las palabras claves. Por lo tanto, tomando intervalos de un tiempo determinado t (o lo faltante hasta el comienzo de otra keyword), se completa el MLF con la palabra reservada FILLER y, de esta manera, se podrán entrenar utilizando el mismo set de datos, los modelos necesarios para cubrir toda la gramática asociada al problema. Veamos un ejemplo donde t es 0.20 segundos. Si "Springfield" es una keyword y las transcripciones originales contienen:

```
8.000 8.200 FAR
8.200 8.700 FROM
8.700 9.645 SPRINGFIELD
9.645 10.250 WHEN
10.250 10.600 THE
```

se convierten en:

```
8.000 8.200 FILLER
8.200 8.400 FILLER
8.400 8.600 FILLER
8.600 8.700 FILLER
8.700 9.645 SPRINGFIELD
9.645 9.845 FILLER
9.845 10.045 FILLER
10.045 10.245 FILLER
10.245 10.445 FILLER
10.445 10.600 FILLER
```

Luego, una vez obtenido el archivo que contiene las transcripciones procesadas, se utiliza la herramienta HLed de HTK que permite convertir este último archivo a uno que contiene transcripciones a nivel fonema de las palabras presentes en el texto. Por ejemplo, si el MLF original contenía `8.700 9.645 SPRINGFIELD` luego del proceso, se transforma en:

```
8.700 8.805 s
8.805 8.910 p
8.910 9.015 r
```

```
9.015 9.120 ih
9.120 9.225 ng
9.225 9.330 f
9.330 9.435 iy
9.435 9.540 l
9.540 9.645 d
```

Cabe aclarar que este proceso toma las palabras y las divide uniformemente entre sus fonemas asignando la misma duración cada una de ellos.

Por otro lado, para comenzar con el entrenamiento, se requieren modelos prototípicos, en particular uno por cada fonema que se utilice y uno para el filler. Para ello, comenzando desde un modelo genérico con topología left-right o ergódica y con la cantidad de estados que corresponda según cómo se configure, se construyen los modelos mencionados usando el comando HCompV.

Finalmente, se necesitan los MFCCs de entrada para el algoritmo de entrenamiento de los modelos. Para ello se utiliza HCopy junto a su configuración típica: ventanas de Hamming con valores de 25 ms de duración tomada cada 10 ms para obtener 39 componentes por ventana que incluyen medición en las frecuencias, energía, deltas y coeficientes de aceleración.

3.2.3. Entrenamiento de modelos

Una vez que se dispone de los datos mencionados en la sección anterior, se procede a estimar los parámetros de los modelos. Al entrenar modelos utilizando el algoritmo forward-backward en HTK, se deben realizar iteraciones luego de plantear un modelo para que este converja. En particular, luego de aumentar la cantidad de mezclas Gaussianas, se recomienda una cierta cantidad de reestimaciones de los modelos.

La herramienta que utilizamos para realizar las reestimaciones es HERest. Por la cantidad de datos de entrenamiento, se utiliza una técnica que consiste en partir el entrenamiento en varias partes y luego juntarlas (split training), esto permitiría paralelizar el cómputo de estos modelos si se necesitara.

El primer paso consta de realizar varias reestimaciones tomando como base el modelo prototípico construido para cada fonema (o unidad básica correspondiente). Luego de tres iteraciones, se procede a agregar modelos de silencio y hacer un ajuste de estos (tie silences)[1].

Luego, cada cierta cantidad de iteraciones de por medio, puede aumentarse la cantidad de mezclas Gaussianas, tanto para las keywords como para los fillers. La herramienta que permite realizar estos aumentos es llamada HHEd en donde el usuario establece una cantidad de mezclas y el sistema incrementa la cantidad en los modelos si es posible[2]. La manera de realizarlo es partiendo la componente más "pesada" hasta llegar al número indicado por la configuración.

La tabla 3.1 muestra un ejemplo típico de ejecución junto con el tiempo de ejecución en cada paso.

[1] Para más detalles ver la sección 3.2.2 (Fixing the Silence Models) del libro de HTK (Young y col. 2006).

[2] Existe un umbral en el peso de las componentes en el cual, las componentes que no lo alcanzan, se descartan (defunct components) (Young y col. 2006).

Tarea	Tiempo	Mezclas
Reestimación	04:48	1
Reestimación	05:05	1
Reestimación	03:12	1
Agregar silencios	00:00	1
Ajustar silencios	00:00	1
Reestimación	02:35	1
Incrementar Mezclas	00:00	3
Reestimación	03:03	3
Reestimación	02:48	3
Reestimación	02:29	3
Reestimación	02:20	3
Incrementar Mezclas	00:00	5
...		

Tab. 3.1: Ejemplo de ejecución en entrenamiento

De esta manera, se consigue el modelo entrenado hasta el punto deseado (y muchos intermedios) que servirán para la etapa final, el testeo de estos modelos.

3.2.4. Testeo de modelos

Para comenzar con el proceso de decodificación encargado de testear nuestros modelos, es necesario contar con el modelo de lenguaje que se utilizará. Para ello, construiremos una gramática para nuestra tarea utilizando el lenguaje provisto por HTK y luego la convertiremos a un grafo con un formato especial también provisto por el sistema.

Las gramáticas en formato HTK son una forma de expresiones regulares extendidas encerradas entre paréntesis. Para indicar la gramática vista en la sección 2.3 (ver figura 2.4), utilizaremos un archivo que contendrá lo siguiente:

```
( SENT-START {FILLER | Keyword1 | Keyword2 | ... | KeywordN } SENT-END )
```

Donde `SENT-START` y `SENT-END` son palabras reservadas que manifiestan el inicio y fin de esta gramática. Las llaves que encierran a las keywords y al filler indican cero o más repeticiones. De esta manera, se adapta a todas las posibles observaciones sobre las que se quiera probar. Luego, utilizaremos la herramienta HParse para construir el grafo equivalente (HTK Wordnet) que necesita el algoritmo de Viterbi para trabajar.

Finalmente, HVite será la herramienta que, dado un conjunto de archivos MFCC y la gramática descripta, determine el camino más probable en nuestros modelos y, por lo tanto, descubra el texto más probable finalizando con la etapa de reconocimiento.

3.2.5. Obtención y métricas de resultados

Una vez que finalizó una prueba, el resultado consiste en un archivo que contiene, para cada grabación, detecciones de todas las keywords con tiempo de inicio y fin junto con un puntaje (una probabilidad) otorgado por el algoritmo de *Viterbi*. Estos resultados serán comparados con la referencia que contiene las transcripciones y de esta manera se podrá calcular qué tan efectivo fue el reconocimiento.

La manera que utilizaremos para calcular resultados será contabilizar la cantidad de aciertos (Hits) y falsas alarmas (FA) que se encuentran en el resultado. Diremos que una

detección res es un Hit si existe, en la referencia, una aparición de keyword ref tal que la keywords es idéntica y el tiempo medio se encuentra entre el tiempo inicial y final de la detección. Es decir, $t_i(res) < t_m(ref) < t_f(res)$ donde t_i, t_m y t_f representan el tiempo inicial, medio y final de una aparición. De otra manera, esa detección constituirá una falsa alarma.

Luego, nos encargaremos de calcular la *curva ROC* (Receiver Operating Characteristic), una medición estándar muy utilizada que grafica la tasa de Hits contra el número de falsas alarmas por keyword por hora (fa/kw-hr). Para reducir esta representación a un único número, se utiliza el *FOM* (Figure of Merit) que equivale a la probabilidad promedio de detecciones correctas de 1 a 10 falsas alarmas por hora (Manos 1996). A continuación la fórmula que utilizamos:

$$FOM = \frac{1}{10t}(\sum_{i=1}^{n} p_i + \alpha p_{n+1})$$

donde

- p_i representa el porcentaje de hits hasta la i-ésima falsa alarma[3];
- t el tiempo en horas de grabación;
- $n = \lceil 10t - 0.5 \rceil$ es decir, aproximadamente 10 por la cantidad de horas de grabaciones;
- $\alpha = 10t - n$ es decir, lo que le falta a $10t - 0.5$ para llegar al entero próximo más cercano.

De esta manera, un FOM cercano a 1 indicará la detección perfecta y en cambio un FOM cercano a cero será un mal indicio de la performance de nuestro reconocedor.

Veamos un ejemplo: dada la siguiente referencia y el resultado obtenido (tomaremos en cuenta sólo una grabación como muestra aunque se puede generalizar directamente para varias grabaciones):

```
(Referencia)                          (Resultado)
0.87  0.96  SPRINGFIELD               1.04 1.11 SUPERINTENDENT -5254.09
1.03  1.11  SUPERINTENDENT           3.12 3.18 OFFICIALS -4494.75
3.12  3.18  OFFICIALS                3.31 3.36 ACCORDING -3523.52
3.31  3.34  ACCORDING                5.60 5.85 MELNICOVE -16622.43
4.97  5.02  SPRINGFIELD              5.89 6.23 SPRINGFIELD -3523.52
5.77  5.84  MELNICOVE               6.25 6.57 MELNICOVE -1224.33
5.89  6.33  CELLPHONE
```

podemos formar la siguiente lista:

1.	SUPERINTENDENT	HIT	-5254.09
2.	OFFICIALS	HIT	-4494.75
3.	ACCORDING	HIT	-3523.52
4.	MELNICOVE	HIT	-16622.43
5.	SPRINGFIELD	FA	-3523.52
6.	MELNICOVE	FA	-1224.33

[3] En caso de no existir la i-ésima falsa alarma, p_i será el porcentaje de hits sobre el total de palabras en la referencia.

Esta lista contiene la detección junto con su resultado y su puntaje asociado[4]. Como se puede ver, es suficiente para calcular cantidad de aciertos y falsas alarmas por keyword.

Para calcular la ROC y el FOM asociados, primero seleccionaremos una keyword en particular y formaremos una lista equivalente a la anterior filtrando solo las apariciones de la keyword seleccionada, juntando los datos de todas las grabaciones, y ordenaremos la lista de forma decreciente según la probabilidad asociada.

Por ejemplo, para el keyword `ACCORDING` podemos obtener una lista con la siguiente información:

1.	ACCORDING	HIT	-3523.52
2.	ACCORDING	FA	-3677.75
3.	ACCORDING	HIT	-4504.52
4.	ACCORDING	HIT	-10029.43
5.	ACCORDING	FA	-16622.52
6.	ACCORDING	HIT	-18622.52
7.	ACCORDING	HIT	-20622.52
8.	ACCORDING	FA	-22622.52

En segundo lugar, una vez ordenada, iremos computando el porcentaje de aciertos hasta un punto de corte seleccionado. Para ello, se divide el número de Hits hasta el punto de corte por el número total de apariciones de la keyword en la referencia. En principio, este punto de corte puede variar en cualquier valor desde justo antes del primer resultado hasta luego del último de la lista. Así, pueden obtenerse $|l|+1$ valores, donde $|l|$ representa el tamaño de la lista.

Por ejemplo, supongamos que la palabra `ACCORDING` aparece 10 veces en la referencia. Si en la lista anterior seleccionamos el punto de corte luego del cuarto valor, el porcentaje de aciertos es 30 %

1.	ACCORDING	HIT	-3523.52
2.	ACCORDING	FA	-3677.75
3.	ACCORDING	HIT	-4504.52
4.	ACCORDING	HIT	-10029.43
5.	ACCORDING	FA	-16622.52
6.	ACCORDING	HIT	-18622.52
7.	ACCORDING	HIT	-20622.52
8.	ACCORDING	FA	-22622.52

Si tomamos todos los posibles valores moviendo el punto de corte y graficamos el porcentaje de Hits en función de la cantidad de falsas alarmas hasta ese punto (dividido la cantidad de horas de testeo), obtenemos la curva ROC. La figura 3.2 muestra la curva ROC construida en base a la tabla anterior suponiendo 90 minutos de testeo.

[4] Este puntaje es el logaritmo de una probabilidad asociada al algorítmo. Para más información ver el libro de Young y col. (2006, pág. 206).

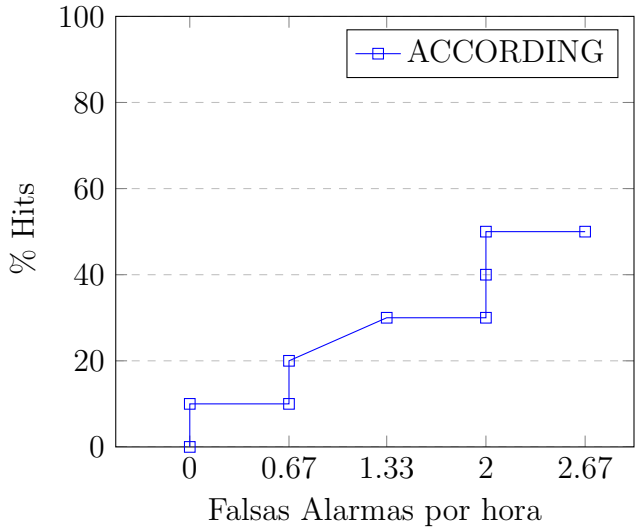


Fig. 3.2: Curva ROC para la palabra ACCORDING

Además, podemos calcular el FOM haciendo:

$t = 1.5$
$n = \lceil 10t - 0.5 \rceil = 15$
$\alpha = 10t - n = 0$
$p_1 = 1/10$
$p_2 = 3/10$
$p_i = 5/10 \quad \text{para } 3 \leq i \leq 15$

$FOM = \frac{1}{10t}(\sum_{i=1}^{n} p_i + \alpha p_{n+1}) = 0.46$

Resumiendo, usaremos dos métricas: la curva ROC, que nos permite visualizar porcentaje de Hits a medida que aumentamos la cantidad de falsas alarmas por hora, y el FOM, que resume las primeras 10 falsas alarmas por hora de la curva ROC promediando dichos valores.

Estas métricas permiten al usuario hacer un balance entre el porcentaje de hits y la cantidad de falsas alarmas que la tarea requiera. En la curva ROC, se ve claramente el efecto en la performance del sistema al tomar como punto de corte los distintos valores del eje x.

4. RESULTADOS

Durante el desarrollo del sistema se realizaron experimentos donde se tomaron varios factores mencionados en capítulos anteriores y se buscó combinarlos con el objetivo de alcanzar un buen rendimiento en el sistema. Las características principales en las que se concentró el estudio fueron:

1. Cantidad de mezclas Gaussianas
2. Intervalo de relleno con fillers
3. Cantidad de estados en el modelo de filler
4. Unidad acústica para el modelo de keywords

A continuación detallaremos cada una de las características evaluadas en los experimentos y presentaremos los resultados obtenidos. Para ello dividimos la sección en tres partes. La primera parte trabaja sobre el conjunto de grabaciones reservadas para desarrollo y busca calibrar la configuración de manera de maximizar el rendimiento del sistema. En la segunda, se verifica que el mejor resultado obtenido en desarrollo, tenga su correlación en el conjunto de control y de esta manera obtener los resultados finales. Además, muestra cómo se ve afectada la performance, dada la inclusión o ausencia de un hablante, durante el entrenamiento utilizando la mejor configuración para el sistema. La tercera parte incluye una discusión sobre los resultados obtenidos.

Cabe destacar que una serie de decisiones fueron tomadas antes de comenzar. Como se discutió en el capítulo 2, hay dos topologías muy utilizadas para HMMs en tareas de reconocimiento; la topología left-right y la topología ergódica. Nuestros experimentos utilizan fillers con topología ergódica y keywords donde cada fonema se representará con un HMM con topología left-right de cinco estados. Además, se decidió trabajar con un conjunto de 20 keywords tomadas del corpus por ser las más frecuentes y tener una longitud mayor a nueve letras. Finalmente, la cantidad de mezclas Gaussianas será incrementada de igual manera tanto para fillers como para keywords.

En la tabla 4.1 puede verse el conjunto de keywords seleccionadas para las pruebas. Las columnas relacionadas con apariciones muestran la cantidad de veces que aparece cada keyword en las transcripciones de referencia.

	Palabra clave	Duración Promedio (seg)	Apariciones	Apariciones en Entenamiento	Apariciones en Testing	Apariciones en Control
1	ACCORDING	0.503	91	50	20	21
2	ADMINISTRATION	0.750	86	67	12	7
3	ASSOCIATION	0.748	50	29	7	14
4	COMMITTEE	0.391	78	57	11	10
5	DEMOCRATIC	0.598	79	46	14	19
6	DEMOCRATS	0.650	48	32	10	6
7	EDUCATION	0.639	64	37	12	15
8	ENVIRONMENTAL	0.702	73	47	11	15
9	GOVERNMENT	0.460	95	58	22	15
10	LAWMAKERS	0.682	44	36	2	6
11	LEGISLATURE	0.692	58	37	10	11
12	MASSACHUSETTS	0.744	457	318	71	68
13	MELNICOVE	0.622	98	65	19	14
14	OFFICIALS	0.486	200	145	25	30
15	POLITICAL	0.541	89	66	12	11
16	PRESIDENT	0.512	82	51	13	18
17	REPUBLICAN	0.637	60	39	8	13
18	SPRINGFIELD	0.676	35	24	7	4
19	SUPERINTENDENT	0.728	43	28	10	5
20	YESTERDAY	0.596	140	99	17	24

Tab. 4.1: Keywords seleccionadas

Cuatro medidas fueron utilizadas para mostrar resultados, estas son:

- FOM promedio sobre todas las keywords
- Promedio ponderado del FOM según la cantidad de apariciones de las keywords
- Porcentaje de Hits sobre todas las keywords[1]
- Cantidad de falsas alarmas detectadas por hora[1]

Cada una de estas mediciones brindará un panorama distinto que, según el objetivo de la tarea de reconocimiento, será más o menos informativo. Por ejemplo, cuando se busca una mayor probabilidad de detección sobre palabras muy frecuentes y no se le da tanto peso al resto de las palabras, la alternativa ponderada de FOM será elegida. En cambio, cuando se busque minimizar falsas alarmas, los gráficos del FOM promedio y falsas alarmas por hora estarán disponibles.

4.1. Resultados sobre el conjunto de desarrollo

En esta sección se presentan las cuatro características principales investigadas. Al finalizar, un breve resumen de estos resultados.

Para estas pruebas, el tiempo total de las keywords en entrenamiento, es decir, la cantidad de tiempo en datos transcriptos, es de aproximadamente 14 minutos. Por otro lado, el tiempo total de grabaciones que se utilizó para el modelo de filler es aproximadamente 7 hs. El testeo se realizó sobre 1.8 horas de grabaciones en donde ocurren 313 apariciones de las keywords.

[1] sin considerar el punto de corte mencionado al final de la sección 3.2.5.

4.1.1. Primera característica: Cantidad de mezclas Gaussianas

Ya que las mezclas Gaussianas determinan las probabilidades de observación, dan flexibilidad en cuanto a reconocer mayor variedad en los vectores acústicos. Es de esperar que modelos con mayor cantidad de mezclas reconozcan con más precisión sonidos similares a los utilizados para el entrenamiento. De todas maneras, un problema que puede surgir es el de sobre-ajuste (overfitting) con respecto a estos datos. Para evitar este problema, se corroborarán los resultados en la siguiente sección.

En los siguientes experimentos, se usarán las componentes de las mezclas como eje x en las mediciones otorgando una visión más amplia para comparar el resto de las decisiones a tomar. En general, el número de mezclas Gaussianas por estado aumentará cada cinco reestimaciones de a dos por incremento variando entre una y al menos 65 componentes.

4.1.2. Segunda característica: Intervalo para fillers

En la sección 3.2.2 hemos visto cómo se completa una transcripción utilizando fillers. Variando el intervalo tendremos fillers que se adaptan a distintas unidades del lenguaje. En nuestros experimentos variamos entre tamaños de fillers que van de 100 a 500 milisegundos. Teniendo en cuenta que 200 ms se acercan al tiempo promedio de una sílaba en inglés (Kent y Forner 1980; Campbell 1992) y 500 ms al de una palabra completa[2].

Los resultados que se muestran a continuación presentan las cuatro métricas mencionadas anteriormente, en forma de tablas y gráficos, donde se pueden ver las mediciones tomadas a medida que la cantidad de mezclas aumenta.

[2] El tiempo promedio de las palabras en este corpus es 0.48 segundos

Mezclas	100ms	200ms	350ms	500ms
1	0.042	0.016	0.003	0.003
3	0.054	0.051	0.003	0.001
5	0.030	0.066	0.005	0.003
7	0.034	0.126	0.006	0.004
9	0.045	0.157	0.007	0.003
11	0.069	0.182	0.001	0.002
13	0.100	0.204	0.014	0.000
15	0.106	0.223	0.011	0.005
17	0.132	0.281	0.007	0.007
19	0.157	0.331	0.016	0.006
21	0.179	0.362	0.016	0.002
23	0.173	0.408	0.041	0.003
25	0.171	0.428	0.046	0.000
27	0.153	0.448	0.038	0.000
29	0.158	0.441	0.043	0.002
31	0.159	0.468	0.071	0.004
33	0.155	0.474	0.050	0.005
35	0.167	0.470	0.066	0.002
37	0.173	0.474	0.071	0.003
39	0.170	0.461	0.080	0.003
41	0.180	0.465	0.079	0.003
43	0.189	0.468	0.074	0.006
45	0.209	0.466	0.073	0.005
47	0.219	0.455	0.085	0.007
49	0.223	0.448	0.093	0.006
51	0.219	0.455	0.101	0.007
53	0.210	0.435	0.119	0.009
55	0.202	0.445	0.117	0.008
57	0.198	0.423	0.126	0.011
59	0.191	0.412	0.131	0.009
61	0.177	0.404	0.141	0.006
63	0.177	0.402	0.126	0.006
65	0.177	0.405	0.125	0.004

Tab. 4.2: Intervalo: FOM promedio

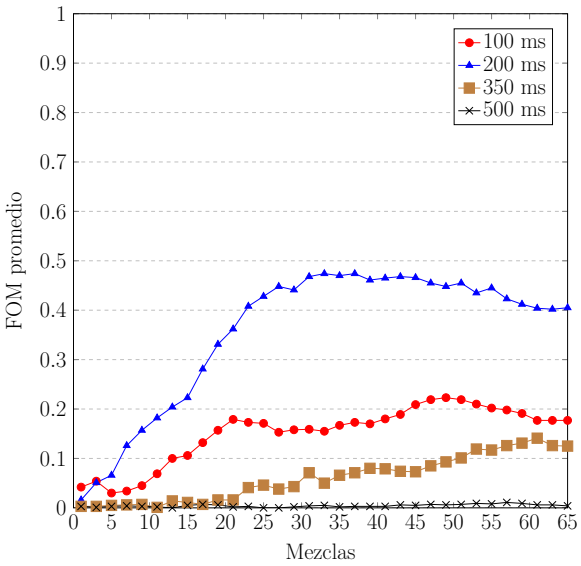


Fig. 4.1: Intervalo: FOM promedio

Puede observarse en esta comparación el gran incremento en el FOM promedio cuando el intervalo se acerca a 200 ms, especialmente a partir de las 25 mezclas. En cambio, los resultados obtenidos para 500 ms no mejoran al aumentar las mezclas de Gaussianas. Por otro lado, los resultados obtenidos para 100 y 350 ms, aunque lejos del óptimo, muestran una tendencia a acercarse a un valor cercano a 0.15.

Mezclas	100ms	200ms	350ms	500ms
1	0.030	0.021	0.004	0.008
3	0.040	0.047	0.002	0.001
5	0.021	0.043	0.005	0.002
7	0.020	0.081	0.006	0.004
9	0.027	0.101	0.006	0.002
11	0.045	0.129	0.001	0.002
13	0.060	0.158	0.009	0.001
15	0.071	0.176	0.007	0.006
17	0.088	0.215	0.005	0.009
19	0.106	0.282	0.011	0.005
21	0.122	0.305	0.014	0.003
23	0.118	0.348	0.021	0.003
25	0.116	0.384	0.021	0.000
27	0.108	0.404	0.023	0.001
29	0.115	0.395	0.032	0.004
31	0.117	0.440	0.046	0.006
33	0.121	0.453	0.037	0.006
35	0.131	0.447	0.046	0.004
37	0.141	0.470	0.048	0.003
39	0.151	0.459	0.048	0.004
41	0.169	0.464	0.054	0.004
43	0.186	0.483	0.054	0.006
45	0.223	0.480	0.051	0.005
47	0.245	0.473	0.056	0.007
49	0.260	0.469	0.071	0.005
51	0.275	0.479	0.066	0.006
53	0.293	0.466	0.088	0.008
55	0.300	0.486	0.090	0.008
57	0.299	0.468	0.099	0.012
59	0.298	0.461	0.098	0.011
61	0.286	0.456	0.111	0.011
63	0.289	0.454	0.100	0.011
65	0.300	0.455	0.099	0.007

Tab. 4.3: Intervalo: FOM ponderado

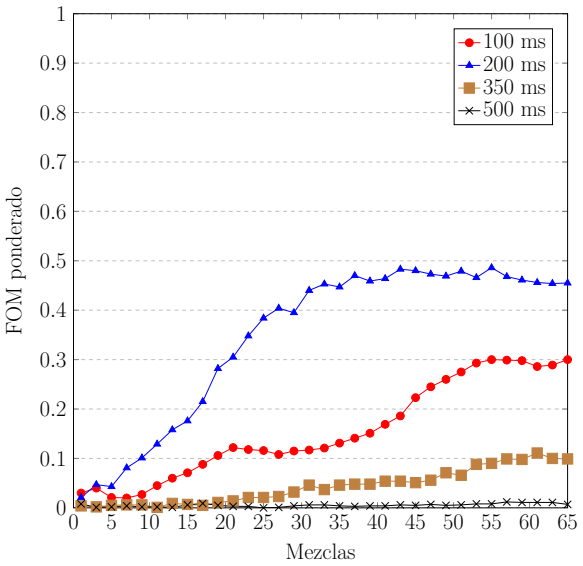


Fig. 4.2: Intervalo: FOM ponderado

Al igual que el resultado anterior, se observa una superioridad del modelo que utiliza 200 ms como intervalo con valores superiores a 0.48. La mayor diferencia con el punto anterior consiste en el alejamiento entre los modelos de 100 y 350 ms. Ya que se está midiendo el FOM ponderado, el resultado indica que las keywords que tienen mayor cantidad de apariciones obtienen mejor resultado en el modelo de 100 ms que en el de 350 ms llegando a picos de 0.3 en el FOM.

Mezclas	100ms	200ms	350ms	500ms
1	63.6	27.5	12.1	10.5
3	57.5	48.2	31.0	22.0
5	59.4	63.9	40.9	28.8
7	66.8	70.9	41.9	32.6
9	66.5	73.2	42.5	32.3
11	67.1	74.4	42.8	31.6
13	70.3	75.4	43.8	30.0
15	68.1	75.1	42.2	29.4
17	63.9	74.4	41.5	30.4
19	61.7	73.5	41.5	29.4
21	60.4	71.9	40.9	28.8
23	59.1	73.8	43.1	27.8
25	57.5	73.8	43.1	27.5
27	56.2	72.5	44.1	26.8
29	55.6	72.2	43.8	27.2
31	54.0	71.6	43.8	27.2
33	52.7	70.6	42.8	26.5
35	53.4	70.0	42.8	26.5
37	52.1	68.7	42.2	27.8
39	50.5	67.1	41.9	26.8
41	50.2	66.8	41.2	26.8
43	49.8	65.5	40.3	26.8
45	49.2	64.5	39.3	25.9
47	48.6	63.6	39.6	26.2
49	48.2	62.9	39.0	25.6
51	47.3	62.6	39.0	25.6
53	45.7	60.7	38.7	25.6
55	45.4	61.3	38.0	24.9
57	44.4	59.1	38.3	25.2
59	43.5	58.1	37.4	24.9
61	42.5	57.5	37.7	24.0
63	42.2	56.9	36.4	23.6
65	41.5	56.5	35.8	23.3

Tab. 4.4: Intervalo: Porcentaje de Hits

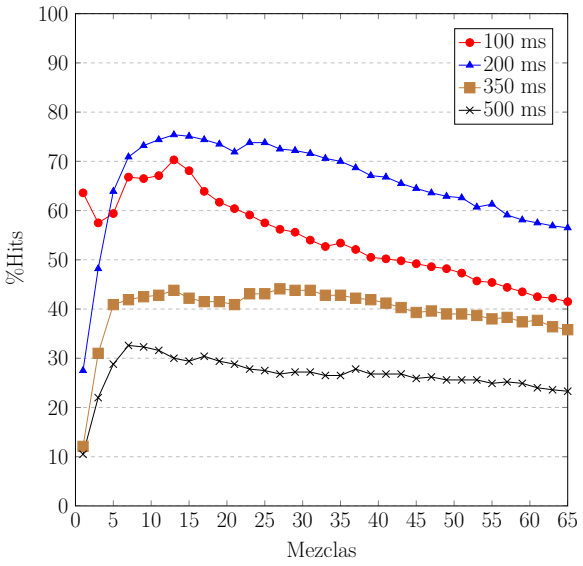


Fig. 4.3: Intervalo: Porcentaje de Hits

En la medición de porcentaje de Hits se ven claramente dos momentos, el de estabilización de los modelos y luego el de refinamiento. Esto quiere decir que hasta aproximadamente diez mezclas en todos los modelos puede verse un incremento abrupto y luego un descenso leve uniforme entre los modelos que se ve acompañado por el resto de los resultados que veremos a continuación. Una vez más, los modelos que utilizan 200 ms mantienen la ventaja de manera constante una vez estabilizados.

Mezclas	100ms	200ms	350ms	500ms
1	4,338.9	3,025.5	2,302.8	2,374.4
3	4,145.9	2,732.1	2,752.6	2,469.2
5	4,025.5	2,121.5	2,856.9	2,789.2
7	3,493.1	1,208.0	2,400.4	2,548.0
9	2,803.1	861.3	2,189.7	2,395.5
11	2,291.7	567.9	1,986.7	2,260.7
13	1,902.9	424.8	1,780.9	2,118.7
15	1,488.6	337.2	1,621.2	2,016.6
17	1,175.8	259.6	1,459.8	1,880.8
19	968.4	206.9	1,295.6	1,765.9
21	814.2	176.9	1,138.1	1,680.0
23	683.9	152.0	1,074.9	1,592.9
25	560.2	136.4	1,024.4	1,491.4
27	465.9	122.0	944.5	1,403.8
29	393.8	109.3	869.1	1,342.2
31	336.7	94.8	794.2	1,271.8
33	282.3	83.2	727.7	1,218.0
35	249.6	79.3	671.1	1,137.5
37	214.1	72.7	619.5	1,078.2
39	181.4	71.0	570.7	1,031.1
41	158.6	67.1	525.8	995.6
43	133.7	62.7	496.9	942.3
45	113.7	59.9	465.9	889.6
47	102.1	57.1	436.5	825.3
49	96.5	56.6	395.5	783.7
51	91.0	53.2	365.5	720.5
53	83.2	52.7	325.0	675.5
55	74.9	50.5	303.4	637.3
57	68.2	48.8	274.0	605.7
59	61.0	46.0	251.8	569.6
61	57.7	46.0	231.3	518.6
63	54.9	45.5	220.2	496.4
65	50.5	43.8	207.4	462.0

Tab. 4.5: Intervalo: Falsas Alarmas por hora

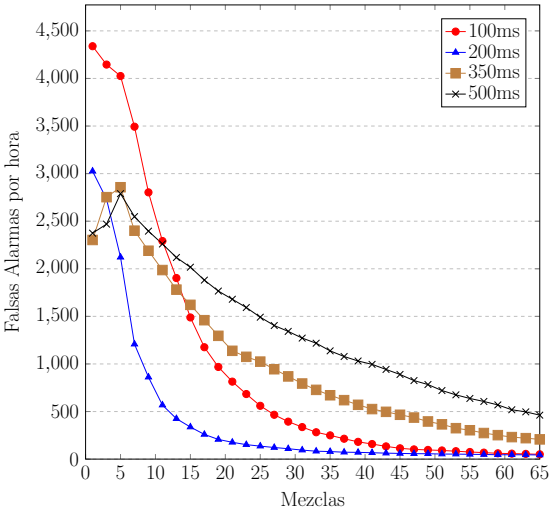


Fig. 4.4: Intervalo: Falsas Alarmas por hora

Las medición de falsas alarmas marca un descenso notable en todos los casos. Cabe aclarar que la mayor parte de estas falsas alarmas son inserciones. Es decir, el modelo de filler no llega a consolidarse hasta luego de estar trabajando con 20 o 30 mezclas en los mejores modelos.

Los resultados anteriores indican una ventaja considerable en los modelos que utilizan fillers cada 200 milisegundos. En las próximas pruebas, trabajaremos con este valor fijo para seguir buscando un máximo según nuestros experimentos.

4.1.3. Tercera característica: Estados en los modelos

A diferencia de los fonemas de keywords que son modelados usando cinco estados, para el filler, variamos entre cinco y siete estados esperando que con siete se logre mayor flexibilidad en los modelos y, por lo tanto, menor cantidad de falsas alarmas.

A continuación, los resultados obtenidos comparando cinco contra siete estados variando la cantidad de mezclas con modelos entrenados utilizando fillers cada 200 milisegundos.

Mezclas	5 estados	7 estados
1	0.000	0.016
3	0.003	0.051
5	0.000	0.066
7	0.022	0.126
9	0.026	0.157
11	0.038	0.182
13	0.062	0.204
15	0.047	0.223
17	0.047	0.281
19	0.061	0.331
21	0.069	0.362
23	0.057	0.408
25	0.055	0.428
27	0.065	0.448
29	0.072	0.441
31	0.083	0.468
33	0.091	0.474
35	0.095	0.470
37	0.105	0.474
39	0.104	0.461
41	0.126	0.465
43	0.131	0.468
45	0.128	0.466
47	0.135	0.455
49	0.139	0.448
51	0.135	0.455
53	0.149	0.435
55	0.146	0.445
57	0.148	0.423
59	0.152	0.412
61	0.145	0.404
63	0.162	0.402
65	0.157	0.405
67	0.156	0.388
69	0.150	0.376
71	0.158	0.370
73	0.163	0.344
75	0.135	0.353
77	0.146	0.344
79	0.158	0.344

Tab. 4.6: Estados: FOM promedio

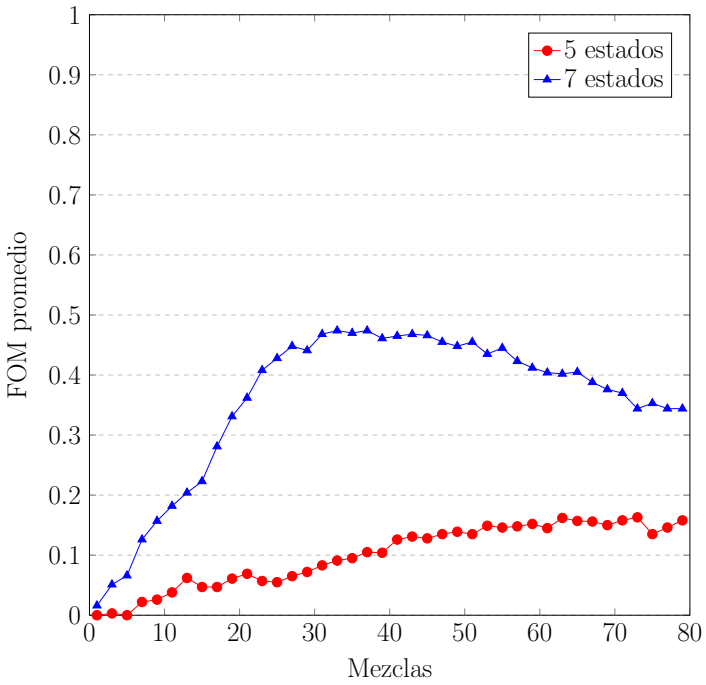


Fig. 4.5: Estados: FOM promedio

El primer resultado muestra una rápida adaptación del modelo de siete estados y un leve progreso en el de cinco con una marcada superioridad en especial de 30 a 40 mezclas Gaussianas.

Mezclas	5 estados	7 estados
1	0.000	0.021
3	0.006	0.047
5	0.000	0.043
7	0.016	0.081
9	0.021	0.101
11	0.026	0.129
13	0.045	0.158
15	0.033	0.176
17	0.031	0.215
19	0.039	0.282
21	0.042	0.305
23	0.040	0.348
25	0.039	0.384
27	0.051	0.404
29	0.054	0.395
31	0.062	0.440
33	0.070	0.453
35	0.079	0.447
37	0.087	0.470
39	0.089	0.459
41	0.108	0.464
43	0.110	0.483
45	0.110	0.480
47	0.117	0.473
49	0.118	0.469
51	0.114	0.479
53	0.128	0.466
55	0.122	0.486
57	0.124	0.468
59	0.126	0.461
61	0.120	0.456
63	0.131	0.454
65	0.129	0.455
67	0.130	0.442
69	0.125	0.433
71	0.134	0.429
73	0.138	0.423
75	0.133	0.429
77	0.146	0.422
79	0.154	0.420

Tab. 4.7: Estados: FOM ponderado

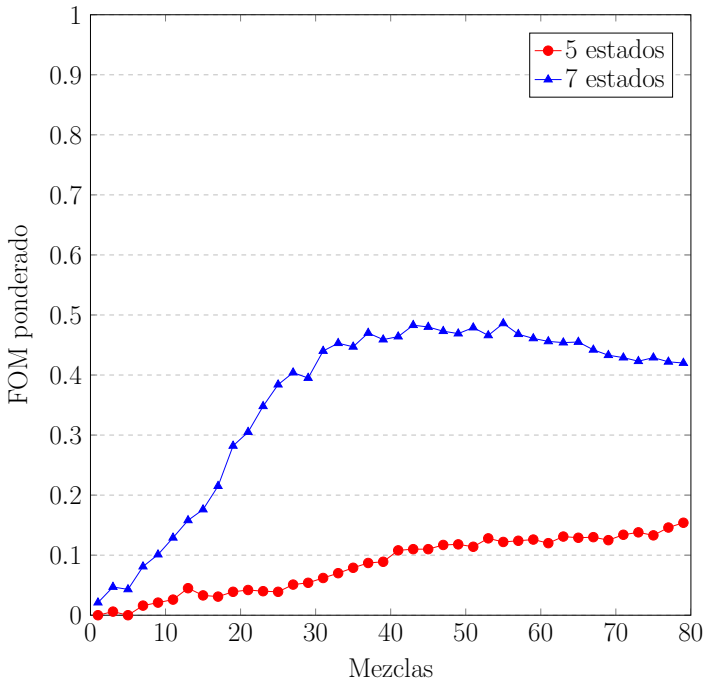


Fig. 4.6: Estados: FOM ponderado

Al igual que el gráfico anterior, se observa la ventaja de utilizar siete estados, sin mayores diferencias a pesar de estar ponderado por la cantidad de apariciones de las keywords. Cabe aclarar que aunque con una mayor cantidad de mezclas las curvas pueden llegar a unirse, el costo computacional es muy alto y por ello consideramos suficiente llegar hasta modelos con 80 mezclas.

Mezclas	5 estados	7 estados
1	16.9	27.5
3	32.6	48.2
5	47.9	63.9
7	50.5	70.9
9	54.6	73.2
11	55.9	74.4
13	61.7	75.4
15	60.1	75.1
17	60.1	74.4
19	61.3	73.5
21	62.3	71.9
23	62.0	73.8
25	61.3	73.8
27	62.3	72.5
29	61.7	72.2
31	62.3	71.6
33	62.6	70.6
35	61.3	70.0
37	61.0	68.7
39	60.1	67.1
41	61.0	66.8
43	59.4	65.5
45	57.5	64.5
47	57.2	63.6
49	57.2	62.9
51	55.6	62.6
53	55.0	60.7
55	54.3	61.3
57	53.7	59.1
59	53.0	58.1
61	50.8	57.5
63	51.4	56.9
65	50.8	56.5
67	49.8	55.3
69	49.2	54.3
71	48.6	54.0
73	48.2	53.4
75	47.6	53.7
77	47.9	52.4
79	48.2	52.4

Tab. 4.8: Estados: Porcentaje de Hits

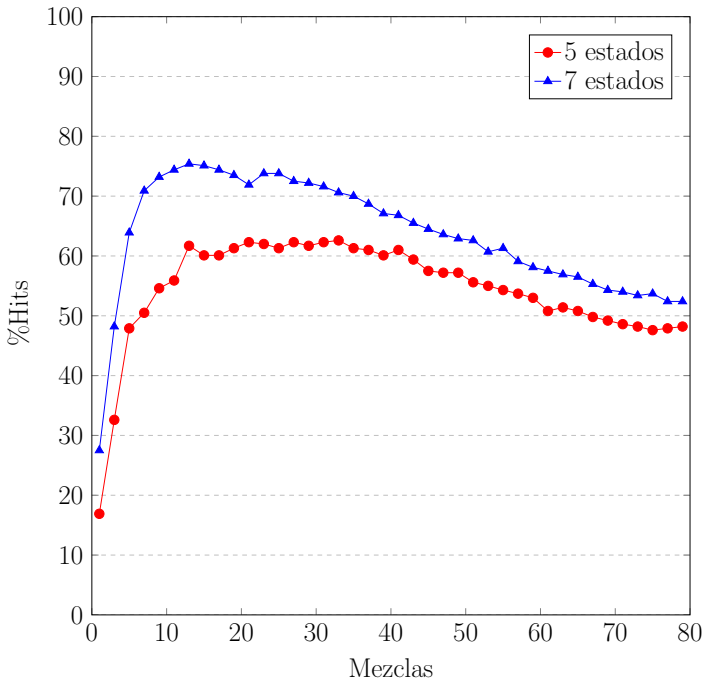


Fig. 4.7: Estados: Porcentaje de Hits

Con respecto al porcentaje de Hits, aunque ambos modelos alcanzan valores entre 60 % y 70 % de aciertos, el modelo de siete estados consigue estar por encima de los mejores resultados para cinco estados.

Mezclas	5 estados	7 estados
1	2,662.8	3,025.5
3	3,078.8	2,732.1
5	2,888.5	2,121.5
7	2,602.3	1,208.0
9	2,443.7	861.3
11	2,339.4	567.9
13	2,203.6	424.8
15	1,989.5	337.2
17	1,868.0	259.6
19	1,771.5	206.9
21	1,627.3	176.9
23	1,534.7	152.0
25	1,473.7	136.4
27	1,377.7	122.0
29	1,303.9	109.3
31	1,220.7	94.8
33	1,157.0	83.2
35	1,078.8	79.3
37	1,033.3	72.7
39	988.4	71.0
41	941.2	67.1
43	875.2	62.7
45	833.1	59.9
47	786.5	57.1
49	763.2	56.6
51	708.3	53.2
53	668.9	52.7
55	618.4	50.5
57	585.1	48.8
59	557.4	46.0
61	527.5	46.0
63	491.4	45.5
65	454.2	43.8
67	432.6	43.3
69	402.1	42.2
71	372.2	41.6
73	345.0	39.9
75	327.2	38.8
77	311.1	37.7
79	297.8	38.8

Tab. 4.9: Estados: Falsas Alarmas por hora

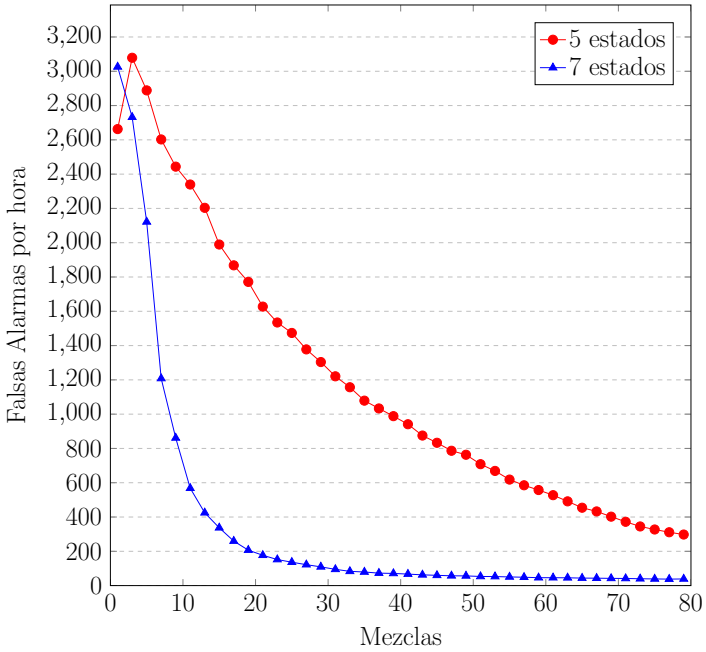


Fig. 4.8: Estados: Falsas Alarmas por hora

En falsas alarmas, el modelo de siete estados muestra obtener resultados aceptables por debajo de 100 a partir de las 30 mezclas. En cambio, el modelo de cinco estados tiene una curva mucho menos pronunciada en donde la cantidad de falsas alarmas supera por mucho a la cantidad de keywords que se desean encontrar. Es claro que la cantidad de inserciones se debe a un modelo menos flexible a diferencia del de siete estados.

En resumen, estos resultados muestran la superioridad de los modelos que utilizan siete estados en todos los aspectos medidos.

4.1.4. Cuarta característica: Unidad acústica

Como se mencionó en la sección 2.4, utilizamos dos alternativas en cuanto a la elección de unidad acústica para una keyword: palabras o fonemas. En el siguiente experimento se esperaba encontrar mejores resultados al utilizar la segunda representación dada la cantidad de datos en el corpus utilizado y las propiedades discutidas de estas alternativas.

Mezclas	Fonema	Palabra
1	0.016	0.000
3	0.051	0.000
5	0.066	0.000
7	0.126	0.000
9	0.157	0.000
11	0.182	0.000
13	0.204	0.000
15	0.223	0.000
17	0.281	0.000
19	0.331	0.000
21	0.362	0.000
23	0.408	0.000
25	0.428	0.000
27	0.448	0.003
29	0.441	0.017
31	0.468	0.027
33	0.474	0.045
35	0.470	0.056
37	0.474	0.070
39	0.461	0.085
41	0.465	0.100
43	0.468	0.093
45	0.466	0.102
47	0.455	0.114
49	0.448	0.111
51	0.455	0.112
53	0.435	0.115
55	0.445	0.115
57	0.423	0.117
59	0.412	0.119
61	0.404	0.119
63	0.402	0.112
65	0.405	0.124

Tab. 4.10: Unidad: FOM promedio

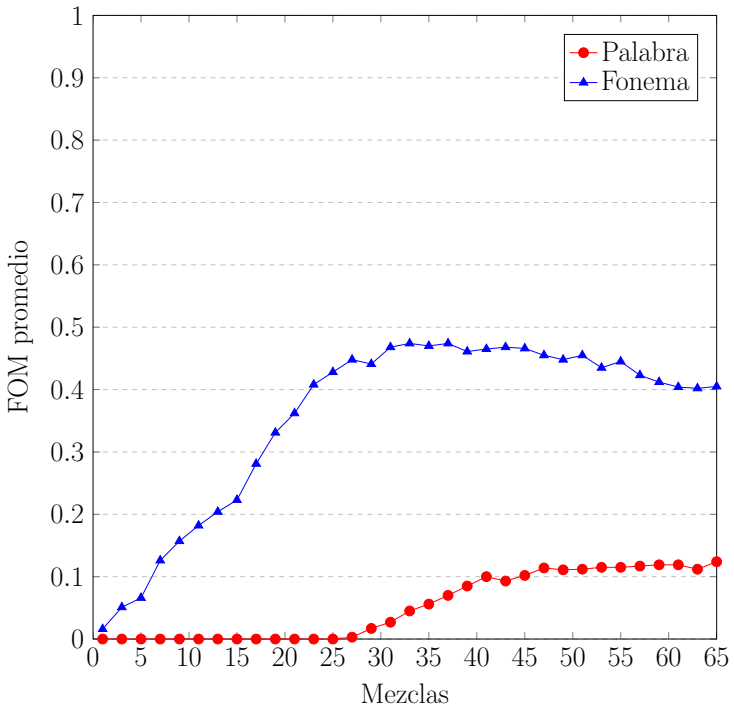


Fig. 4.9: Unidad: FOM promedio

En efecto, el FOM promedio utilizando fonemas como unidad acústica supera en gran medida a los modelos compuestos de palabras completas. Por esta razón, no mostraremos el resto de las visiones ya que la diferencia es muy grande entre estas alternativas.

En este punto, cabe destacar que falta mucha exploración para descartar que las palabras sean una mejor unidad. El modelo que utilizamos fue un HMM left-right de 5 estados por palabra.

4.1.5. Resumen del desarrollo

De este primer estudio obtenemos una clase de modelos que maximiza nuestras mediciones, a continuación las características:

- intervalo para completar con fillers de 200 milisegundos;
- modelo de filler con siete estados (cinco emisores);
- modelos de keywords utilizando fonemas como unidad mínima.

En particular, el mejor resultado (según el FOM) fue devuelto por el modelo con 37 mezclas de Gaussianas que puede observarse en la tabla 4.11. Esta tabla muestra los números alcanzados por el modelo.

Mezclas	FOM promedio	FOM ponderado	% Hits	FA/h	Apariciones
37	0.474	0.47	68.7	72.7	313

Tab. 4.11: Mejor resultado en desarrollo

4.2. Pruebas Finales

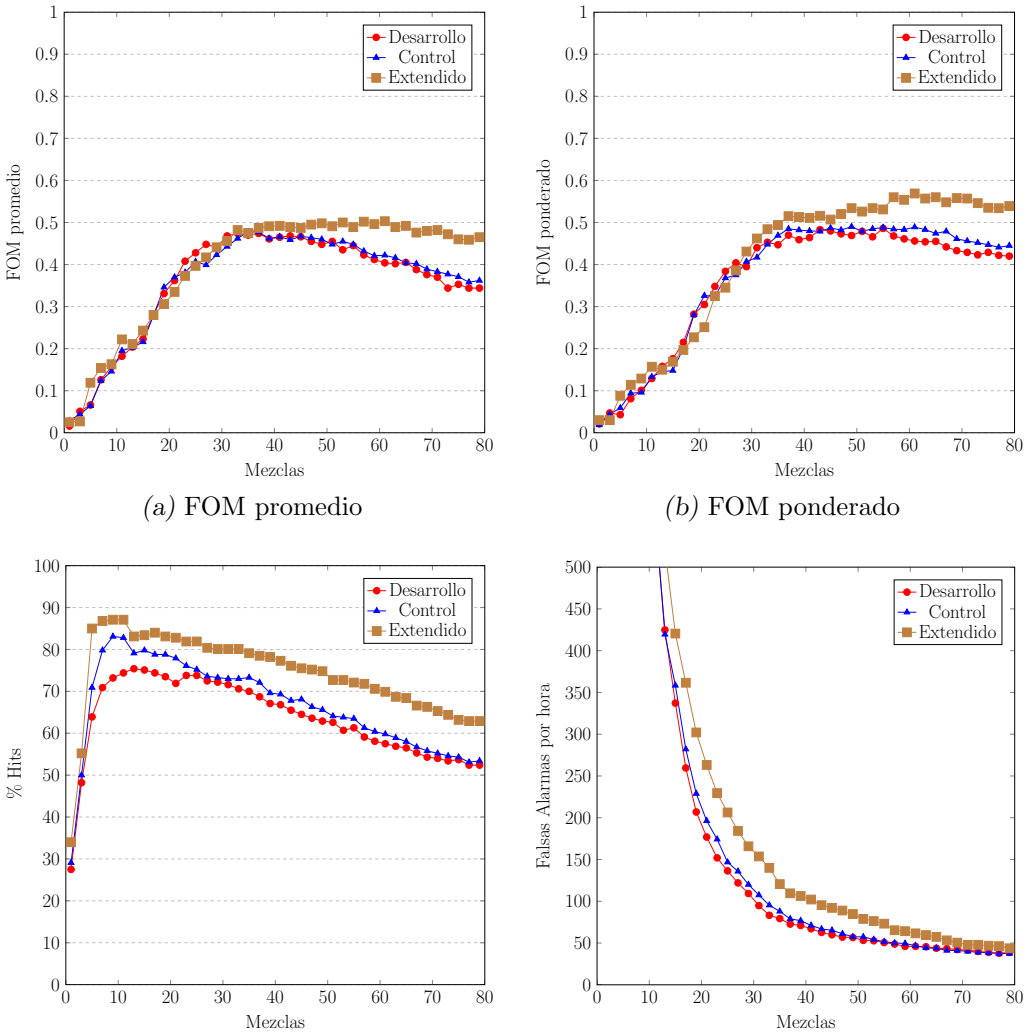
Utilizando la configuración que permitió obtener los mejores modelos en los pasos anteriores, se realizaron dos pruebas finales: la primera, sobre el conjunto de datos reservado para control y la segunda dejando fuera un hablante por vez en el entrenamiento para estudiar la flexibilidad del sistema ante estos cambios.

4.2.1. Prueba sobre set de control

Para las siguientes pruebas se utilizó la sexta parte del corpus reservada para este propósito. Utilizando los mejores modelos entrenados para las pruebas en desarrollo, se testeó sobre esta reserva buscando obtener resultados más realistas por tratarse de datos no utilizados hasta el momento.

Además, como prueba secundaria, se utilizó la misma configuración que produjo los mejores resultados en desarrollo y se entrenaron nuevos modelos sobre los 5/6 del corpus utilizados en desarrollo (1/5 más que el resto de los resultados). Llamaremos a esta prueba: "resultado extendido".

Las pruebas se realizaron sobre 1.9 hs de grabaciones conteniendo un total de 326 apariciones de keywords. A continuación los resultados obtenidos comparándolos con la misma configuración en el conjunto de desarrollo.



(a) FOM promedio

(b) FOM ponderado

(c) % Hits

(d) Falsas Alarmas

Fig. 4.10: Resultado final: Prueba control

Los cuatro resultados anteriores muestran un comportamiento casi idéntico de los mejores modelos cuando se testeó en los datos de desarrollo y en los datos de control. Por otro lado, se ve un incremento en la performance en la prueba extendida en todos los aspectos, excepto en falsas alarmas. Este incremento es más notorio a partir de las 40 mezclas en el caso del FOM. Además, el porcentaje de hits se mantiene siempre arriba del resto acompañado por un mayor número de falsas alarmas por hora.

En la tabla 4.12 se comparan los resultados para estos modelos entre. Las tablas 4.13, 4.14 y 4.15 muestran el detalle de estos resultados discriminando por keyword utilizando la misma cantidad de mezclas que el mejor resultado en desarrollo.

Ambiente	Mezclas	FOM promedio	FOM ponderado	% Hits	FA/h	Apariciones
Desarrollo	37	0.474	0.470	68.7	72.7	313
Control	37	0.477	0.485	72.1	78.9	326
Extendido	37	0.487	0.515	78.5	109.5	326

Tab. 4.12: Mejor resultado en desarrollo comparado

	Keyword	% Hits	FA	FOM	Sustituciones	Inserciones	Apariciones
1	ACCORDING	35.0	4	0.306	0	4	20
2	ADMINISTRATION	91.7	5	0.686	0	5	12
3	ASSOCIATION	57.1	0	0.571	0	0	7
4	COMMITTEE	45.5	11	0.409	0	11	11
5	DEMOCRATIC	50.0	0	0.5	0	0	14
6	DEMOCRATS	100.0	13	0.578	7	6	10
7	EDUCATION	83.3	14	0.306	1	13	12
8	ENVIRONMENTAL	54.5	4	0.505	0	4	11
9	GOVERNMENT	22.7	1	0.22	0	1	22
10	LAWMAKERS	100.0	1	0.945	0	1	2
11	LEGISLATURE	90.0	5	0.684	0	5	10
12	MASSACHUSETTS	91.5	8	0.587	0	8	71
13	MELNICOVE	84.2	38	0.053	0	38	19
14	OFFICIALS	88.0	12	0.51	0	12	25
15	POLITICAL	75.0	3	0.722	0	3	12
16	PRESIDENT	53.8	7	0.419	0	7	13
17	REPUBLICAN	25.0	3	0.222	0	3	8
18	SPRINGFIELD	28.6	0	0.286	0	0	7
19	SUPERINTENDENT	20.0	0	0.2	0	0	10
20	YESTERDAY	82.4	2	0.778	0	2	17
	Resumen	68.7	131	0.474	8	123	313

Tab. 4.13: Resultado en desarrollo por keyword (37 mezclas sobre 1.8 hs de grabaciones)

	Keyword	% Hits	FA	FOM	Sustituciones	Inserciones	Apariciones
1	ACCORDING	28.6	1	0.278	0	1	21
2	ADMINISTRATION	85.7	8	0.572	0	8	7
3	ASSOCIATION	50.0	2	0.459	0	2	14
4	COMMITTEE	70.0	8	0.663	0	8	10
5	DEMOCRATIC	52.6	2	0.499	0	2	19
6	DEMOCRATS	100.0	16	0.518	6	10	6
7	EDUCATION	73.3	11	0.408	0	11	15
8	ENVIRONMENTAL	60.0	1	0.568	0	1	15
9	GOVERNMENT	26.7	1	0.263	0	1	15
10	LAWMAKERS	83.3	4	0.702	0	4	6
11	LEGISLATURE	81.8	5	0.608	0	5	11
12	MASSACHUSETTS	89.7	8	0.606	0	8	68
13	MELNICOVE	100.0	42	0.0	0	42	14
14	OFFICIALS	93.3	20	0.283	0	20	30
15	POLITICAL	72.7	4	0.679	0	4	11
16	PRESIDENT	88.9	9	0.629	0	9	18
17	REPUBLICAN	23.1	4	0.186	0	4	13
18	SPRINGFIELD	25.0	0	0.25	0	0	4
19	SUPERINTENDENT	60.0	0	0.6	0	0	5
20	YESTERDAY	87.5	4	0.763	0	4	24
	Resumen	72.1	150	0.477	6	144	326

Tab. 4.14: Resultado control por keyword (37 mezclas sobre 1.9 hs de grabaciones)

	Keyword	% Hits	FA	FOM	Sustituciones	Inserciones	Apariciones
1	ACCORDING	61.9	8	0.466	0	8	21
2	ADMINISTRATION	85.7	7	0.662	0	7	7
3	ASSOCIATION	64.3	1	0.617	0	1	14
4	COMMITTEE	50.0	21	0.321	0	21	10
5	DEMOCRATIC	63.2	3	0.554	0	3	19
6	DEMOCRATS	100.0	18	0.492	6	12	6
7	EDUCATION	80.0	21	0.022	0	21	15
8	ENVIRONMENTAL	66.7	2	0.614	0	2	15
9	GOVERNMENT	80.0	8	0.625	0	8	15
10	LAWMAKERS	83.3	5	0.684	0	5	6
11	LEGISLATURE	81.8	6	0.589	0	6	11
12	MASSACHUSETTS	89.7	6	0.651	0	6	68
13	MELNICOVE	100.0	50	0.0	0	50	14
14	OFFICIALS	90.0	19	0.418	0	19	30
15	POLITICAL	63.6	11	0.445	0	11	11
16	PRESIDENT	94.4	10	0.661	0	10	18
17	REPUBLICAN	53.8	7	0.425	0	7	13
18	SPRINGFIELD	25.0	0	0.25	0	0	4
19	SUPERINTENDENT	60.0	0	0.6	0	0	5
20	YESTERDAY	83.3	5	0.652	0	5	24
	Resumen	78.5	208	0.487	6	202	326

Tab. 4.15: Resultado extendido por keyword (37 mezclas sobre 1.9 hs de grabaciones)

Estas tablas muestran una gran similitud entre los resultados en cuanto a FOM, por otra parte, las pruebas sobre el set de datos de control obtuvo un mayor porcentaje de Hits acompañado de una mayor cantidad de falsas alarmas por hora al igual que el resultado extendido.

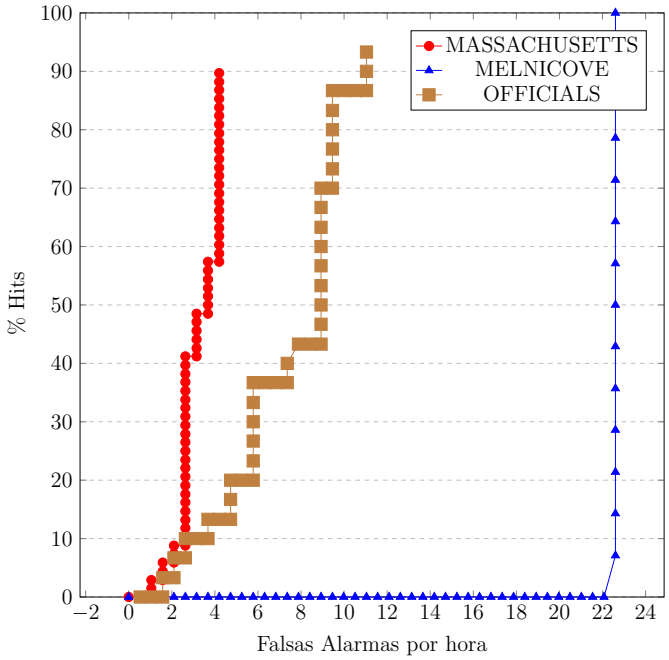


Fig. 4.11: Curva ROC para tres keywords

La figura anterior muestra la curva ROC para tres keywords elegidas por sus diferencias. Puede verse en el eje x la cantidad de falsas alarmas por hora y el impacto que podría producir en el porcentaje de Hits optar por aceptar esa cantidad como umbral de

detección para el sistema. En el caso de `MASSACHUSETTS` permitir 5 falsas alarmas por hora produciría un porcentaje de Hits máximo, en cambio, para `MELNICOVE` impactaría dejando un pésimo resultado.

4.2.2. Prueba por Hablante

El siguiente experimento midió la performance del sistema cuando se entrena con todos los locutores salvo un individuo y luego se prueba sobre esa persona. Para ello, se fue tomando cada una de las siete personas presentes en el corpus y se entrenó con el resto del grupo, luego se probó sobre el sujeto elegido y se tomaron los resultados. Este proceso se repitió por cada locutor, recolectando las detecciones en una sola tabla sobre la cual se calcula la performance global.

Hablante	FOM promedio	FOM ponderado	% Hits	Falsas Alarmas por hora	Apariciones
Mujer 1	0.335	0.527	64.8	46.3	267
Mujer 2	0.351	0.394	46.9	21.9	224
Mujer 3	0.253	0.301	49.3	127.4	554
Hombre 1	0.034	0.05	05.1	27.8	158
Hombre 2	0.126	0.305	33.5	19.7	155
Hombre 3	0.473	0.433	48.5	21.9	101
Hombre 4	0.126	0.189	21.7	10.1	511
Global	0.102	0.105	39.1	47.2	1970

Tab. 4.16: Resultados por hablante

La tabla 4.16 muestra el detalle por hablante. En estos resultados se observa una degradación de la performance del sistema en general, pero se ve que para ciertas personas funcionó al mismo nivel que los mejores resultados obtenidos en desarrollo, como los números en el sujeto Hombre 3 con 0.473 de FOM promedio. También se puede ver un gran deterioro en el porcentaje de hits y, contrarrestando el mal resultado, una caída positiva en la cantidad de falsas alarmas por hora.

4.3. Discusión

Durante la etapa de desarrollo, no se vieron mayores diferencias entre el FOM ponderado y el FOM promedio. Este hecho puede indicar que la performance del sistema no varía demasiado ante keywords con muchas ocurrencias ya que el entrenamiento provisto por estas aporta al resto de los modelos.

En la comparación de palabras contra fonemas como unidad mínima se notó una gran ventaja al trabajar con fonemas. Esto puede deberse a la cantidad de estados utilizados para representar palabras completas ya que al incrementar considerablemente la cantidad de mezclas en los modelos comienza a verse un incremento en las mediciones. Es posible que utilizando una cantidad de estados variable dependiendo de la longitud de la palabra se consiga una mejor performance.

En la sección 4.2, la prueba de control muestra que el mejor modelo alcanza rendimientos superiores a 0.47 en FOM promedio, 72.1 % de Hits y 78.9 falsas alarmas por hora (con 326 apariciones de keywords totales en 1.9 horas de grabaciones). Yendo al detalle de las falsas alarmas en esta prueba, puede verse que sólo seis son sustituciones y todas ocurrieron en el caso de la palabra DEMOCRATS, en el cual todas fueron producidas por detectar la palabra DEMOCRATIC en su lugar. Esto parece indicar que los modelos de fonemas se adaptaron con gran precisión a los fonemas presentes en el entrenamiento. En

cambio, faltó refinar el modelo de fillers para evitar falsas detecciones en momentos en los que no aparece ninguna keyword.

Por otra parte, se alcanzó un 82 % de Hits para 11 mezclas en la prueba de control. Esto muestra que si el objetivo es realizar un sistema con probabilidad alta de encontrar palabras claves, sin darle tanta importancia a las falsas alarmas, se puede utilizar dicha configuración.

La prueba extendida, muestra que ante más datos de entrenamiento, el sistema responde variando su performance de manera notable. Esto último hace pensar que sólo 15 minutos de datos transcriptos son insuficientes para que el sistema llegue a su mejor rendimiento.

Finalmente, en la prueba por hablante, puede verse una caída notable en el promedio del FOM. Estos resultados son de esperar dado que no se puede generalizar a nuevos hablantes habiendo entrenado con sólo seis. Para evaluar los resultados, hicimos una consulta informal a una estudiante de fonoaudiología en donde le pedimos que indique similitudes y diferencias entre las voces presentes y también características relevantes que pueda detectar. El resultado indicó que la voz de las tres mujeres es bastante similar entre sí y, además, la forma de hablar de la mujer 1 con la del hombre 3 tienen mucho en común. Por otra parte, la voz del hombre 1 parece ser muy monótona comparada con el resto. Esta información se ajusta con precisión a los resultados obtenidos, ya que el hombre 1 está muy por debajo del resto y que los resultados del hombre 3 y la mujer 2 están entre los mejores del grupo (FOM promedio 0.335 y 0.473). Por otro lado, las falsas alarmas por hora en la detección fueron menores a las obtenidas en las pruebas anteriores, lo que da a pensar que faltó entrenamiento para el modelo de keywords para lograr un buen funcionamiento ante la diferencia entre los hablantes.

En cuanto a comparaciones con otros trabajos, no hemos encontrado artículos comparables de manera directa dadas las características del sistema, especialmente por el tiempo de entrenamiento. Para hacer una comparación aproximada, podemos tomar el artículo de Garcia y Gish (2006) como referencia. En este trabajo se realiza experimentos sobre siete horas de grabaciones donde se encuentran 30 hablantes. En estas grabaciones busca seis palabras claves en español habiendo entrenado el sistema con 15 minutos de datos transcriptos (y 2 horas de datos no transcriptos). Los resultados obtenidos fueron cercanos a 0.35 en FOM promedio. De todas maneras, cabe aclarar que en los 15 minutos de entrenamiento se disponía de una cantidad muy pequeña de repeticiones de las keywords a buscar.

Otro trabajo interesante es el de Jansen y Niyogi (2009), en donde explican que no existen muchos trabajos con el detalle necesario para poder comparar resultados y, por lo tanto, se encargan de documentar una posible base de referencia sobre el tema. El modelo que utilizan es el de un filler que contiene un modelo por cada uno de los posibles fonemas del lenguaje y keywords en donde cada fonema es representado con un solo estado. Para entrenar los modelos utilizan la base de datos TIMIT (Garofolo y col. 1993) y al testear sobre el corpus de Boston (Ostendorf, Price y Shattuck-Hufnagel 1995) llegan a un FOM promedio de 0.581.

5. CONCLUSIONES

En esta tesis se construyó y estudió un sistema para detectar palabras claves en habla continua con una cantidad mínima de datos de entrenamiento. Una serie de técnicas fueron utilizadas para modelar un conjunto de keywords predefinidas utilizando Modelos Ocultos de Markov junto con un modelo de filler para reconocer el resto de los sonidos presentes en las grabaciones. Luego, utilizando el Boston University Radio Speech Corpus se entrenaron diversos modelos que permitieron la búsqueda de la mejor configuración para luego estudiar las alternativas. Los puntos principales fueron:

- el estudio de la bibliografía en búsqueda de técnicas para modelar el problema de reconocimiento de palabras claves;
- la construcción de un sistema sobre HTK que permita resolver el problema;
- el entrenamiento de modelos y la detección de la configuración óptima variando la unidad acústica, la cantidad de estados, la cantidad de mezclas Gaussianas y el intervalo para completar transcripciones en el entrenamiento de fillers;
- la evaluación de resultados sobre un conjunto del corpus reservado para dicha tarea y una versión en la cual se testea sobre hablantes no presentes en el entrenamiento.

Luego del estudio, podemos concluir que es posible realizar un sistema con las características planteadas con un rendimiento aceptable y mejorable, sin la necesidad de incluir técnicas muy complejas ni grandes cantidades de datos transcriptos. Si se quisiera producir un sistema con las características que hemos estudiado, se necesita un conjunto de grabaciones sin transcripciones (siete horas aproximadamente) con calidad similar a la del corpus[1] y alrededor de 15 minutos de grabaciones que contengan las palabras claves que se desee encontrar junto con su transcripción[2] a nivel de palabras.

Por otra parte, este estudio deja abierta una serie de preguntas sin responder relacionadas principalmente con la búsqueda de mejores resultados. Alguno de los caminos posibles a seguir son:

- ver cómo afecta al sistema la incorporación de nuevos datos de entrenamiento, ya sea mediante más repeticiones o mediante nuevas keywords que deseen agregarse;
- desarrollar una implementación de modelos de fillers más complejos manteniendo la restricción de no tener datos transcriptos;
- buscar nuevas combinaciones de cantidad de estados y mezclas Gaussianas que mejoren el rendimiento;
- estudiar el impacto de variar factores como pesos en la gramática o penalidad de inserción de palabras en el algoritmo de Viterbi;
- mejorar el sistema para hacerlo más resistente a cambios de hablante.

[1] ver 3.1 para detalles en la calidad del audio.
[2] ver 3.2.2 para más detalles.

BIBLIOGRAFíA

Bridle, JOHN S (1973). "An efficient elastic-template method for detecting given words in running speech". En: *Brit. Acoust. Soc. Meeting*.

Campbell, W Nick (1992). "Syllable-based segmental duration". En: *Talking machines: Theories, models, and designs*, págs. 211-224.

Das, Samarjit y Guwahati India (2005). *Speaker Dependent Bengali Keyword Spotting in Unconstrained english speech*.

Garcia, Alvin y Herbert Gish (2006). "Keyword spotting of arbitrary words using minimal speech resources". En: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 1. IEEE.

Garofolo, JS y col. (1993). *TIMIT Acoustic-Phonetic Continuous Speech Corpus, Linguistic Data Consortium, Philadelphia*.

Jansen, A y P Niyogi (2009). *An experimental evaluation of keyword-filler Hidden Markov Models*. Inf. téc. Technical report, University of Chicago.

Jurafsky, Dan y James H Martin (2009). *Speech & Language Processing*. 2nd edition. Prentice Hall.

Kent, Raymond D y LL Forner (1980). "Speech segment duration in sentence recitations by children and adults." En: *Journal of Phonetics*.

Manos, Alexandros Sterios (1996). "A study on out-of-vocabulary word modelling for a segment-based keyword spotting system". Tesis doct. Massachusetts Institute of Technology.

Moore, Roger K (2003). "A comparison of the data requirements of automatic speech recognition systems and human listeners." En: *INTERSPEECH*.

Ostendorf, Mari, PJ Price y Stefanie Shattuck-Hufnagel (1995). "The Boston University radio news corpus". En: *Linguistic Data Consortium*.

Rabiner, Lawrence (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". En: *Proceedings of the IEEE* 77.2.

Rohlicek, J Robin y col. (1989). "Continuous hidden Markov modeling for speaker-independent word spotting". En: *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. IEEE.

Rose, Richard C y Douglas B Paul (1990). "A hidden Markov model based keyword recognition system". En: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE.

Thambiratnam, Albert JK (2005). "Acoustic keyword spotting in speech with applications to data mining". Tesis doct. Queensland University of Technology.

Weintraub, Mitchel (1993). "Keyword-spotting using SRI's DECIPHER large-vocabulary speech-recognition system". En: *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*. Vol. 2. IEEE.

Young, S y col. (2006). "The HTK-Book 3.4". En: *Cambridge University, Cambridge, England*.