RECUPERACIÓN AUTOMÁTICA DE INFORMACIÓN EN DOCUMENTOS DE AUDIO MEDIANTE UNA ARQUITECTURA DISTRIBUIDA

Autor: Raquel Malo González

Director: Alfonso Ortega Giménez

Dpto de Ingeniería Electrónica y Comunicaciones

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

INGENIERÍA DE TELECOMUNICACIONES

CURSO 2011-2012

Marzo de 2012

Gracias a don Alfonso Ortega, sin cuya orientación y ayuda no habría podido llevar a cabo este proyecto.

A mis compañeros de carrera, por los buenos momentos vividos, las dudas resueltas y el trabajo en equipo.

A Quique, desarrollador de la arquitectura, sin él no habría avanzado tan pronto cuando surgieron los problemas.

Y por último, gracias a mis amigos y a mis padres por confiar siempre en mí y mostrarme su apoyo incondicional.

RECUPERACIÓN AUTOMÁTICA DE INFORMACIÓN EN DOCUMENTOS DE AUDIO MEDIANTE UNA ARQUITECTURA DISTRIBUIDA.

RESUMEN

Gran parte de los documentos de origen reciente (entrevistas, declaraciones, discursos, etc.) que se encuentran en la red tienen un formato multimedia y, por su importancia y relevancia, deben ser preservados y puestos a disposición de los investigadores, los estudiosos y la ciudadanía en general.

El objetivo de este proyecto ha sido dotar de accesibilidad a estos grandes archivos multimedia, ya que, hoy por hoy, el acceso a determinadas grabaciones se convierte en una labor larga y tediosa y, en multitud de ocasiones, el acceso al fragmento específico de la grabación es en la práctica imposible.

Uno de estos ejemplos, en el que nos hemos basado, es el fondo documental de Aragón Radio, puesto a disposición del grupo de investigación para su uso con fines científicos y educativos. El acceso a un fragmento concreto de forma sencilla requiere de un etiquetado automático de cada archivo de audio con su contenido. Para la indexación de la información, se cuenta con una serie de herramientas previamente desarrolladas que permiten la segmentación y clasificación de los archivos de audio, la distinción de los fragmentos de voz de los de música o silencio y la transcripción automática que proporciona las correspondientes etiquetas con el texto asociado. A partir de ahí, la Base de Datos queda conformada por un conjunto de archivos de lenguajes de marcas, que contienen el texto y las etiquetas correspondientes vinculadas al documento de audio.

El proyecto consta de dos partes: la primera de ellas es el desarrollo de la aplicación que permite al usuario recuperar los documentos indexados a través de un motor de búsqueda (MG [1]), que se comunica con el usuario mediante una interfaz web. La segunda parte avanza en la creación de la plataforma que permite obtener de forma automática las etiquetas relevantes a partir del audio, realizando así la ingesta de nuevos documentos al archivo. Todo ello se ha llevado a cabo haciendo uso de una arquitectura distribuida (EDECAN [2]) para dotar de la máxima versatilidad al sistema final.

ÍNDICE DE CONTENIDOS

1 Introducción

Ante la gran cantidad de material multimedia y la expansión de la información disponible, así como de los avances tecnológicos incluyendo el rápido crecimiento de la capacidad de almacenamiento de datos, la necesidad de una búsqueda y recuperación automática y eficiente de la información ha llegado a ser muy importante.

Hoy en día, son admirables los avances en la recuperación de información basada en texto y numerosos motores de búsqueda comerciales están accesibles online, permitiendo encontrar el artículo favorito dentro de una vasta colección. Pero cada vez más, la demanda y el interés se centran en tareas más complicadas que permitan recuperar de forma automática la información de los abundantes archivos multimedia: audios, imágenes, vídeos, música.

La rápida expansión de las colecciones multimedia incluye grabaciones de documentos hablados, emisiones de radio y emisiones de televisión (este proyecto se centra en archivos de voz: radio y documentos hablados). Por eso los estudios actuales centran su atención en el campo de búsqueda conocido como *Spoken Document Retrieval* (SDR [3]) o recuperación de documentos hablados (Figura 1.1). Su importancia es clara. Entre los motivos de su desarrollo se encuentran: (i) los documentos hablados contienen más información además de la mera transcripción, (ii) la necesidad de tratar la amplia información multimedia disponible, ya que cada vez más, toda la información accesible es de este tipo, (iii) método de búsqueda eficiente para poder navegar por un corpus de discursos de gran tamaño tales como los contenidos en el *Digital Voice Library* (DVL), emisiones de noticias, lecturas y archivos encontrados, y (iv) el progreso reciente en tecnologías de procesado y reconocimiento de voz han permitido el uso de indexación automática de documentos hablados. [4]

No obstante, pese a los avances en este campo, la complejidad de la tarea hace que todavía quede camino por recorrer hasta conseguir la finura obtenida en la recuperación de textos. Además, siempre tendremos la restricción en el tamaño de las bases de datos, ahora mucho mayores que si solo se tratase de archivos de texto.

1.1 Objetivos generales

El objetivo principal de este proyecto ha sido diseñar un sistema de búsqueda para los archivos de voz que disponíamos (en un principio el fondo documental de Aragón Radio, aunque puede ser extensible a otros córpora). Este sistema se basa en un software modular (EDECAN [2]) al que se pueden añadir o quitar componentes, o incluso cambiarlos por otros. Esta modularidad le da versatilidad al diseño y permite cómodamente añadir módulos funcionales que tengamos ya diseñados para el tratamiento del audio.

El diseño de este sistema de búsqueda se ha estructurado en dos partes: la primera en el desarrollo de una interfaz web que permite la interacción con el usuario, y gracias

a un motor de búsqueda (MG [1]), devuelve de la base de datos la información correspondiente a la petición del usuario. La segunda, es el tratamiento o análisis del audio (voz) necesario para realizar la indexación que buscamos.

Esta segunda parte es en realidad la primera fase en la construcción del sistema. Consta a su vez de varios módulos representados por la segmentación del audio, la generación de la transcripción y la construcción de metadatos. Finalizada esta primera fase (inscripción o ingesta del audio), el material de audio estará disponible a través del motor de búsqueda online de la parte I.

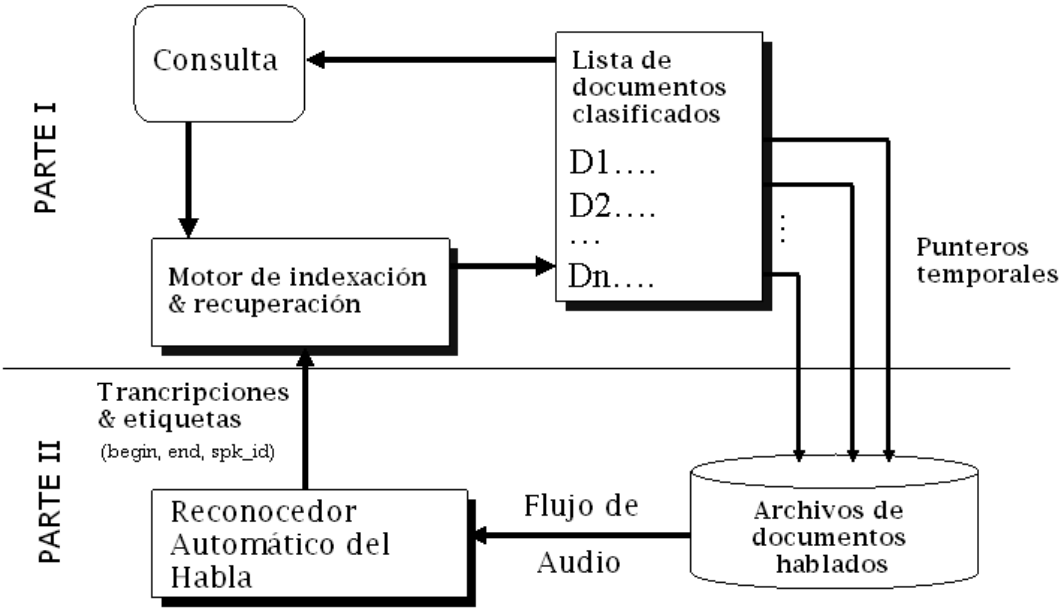

Figura 1.1: *Esquema de un sistema SDR completo*

Para el tratamiento de la voz en la parte II del sistema, se integran varios módulos necesarios para conseguir el resultado final (transcripción más etiquetas). Son el detector de actividad vocal, el segmentador y el reconocedor. El detector de voz es un módulo previo a la segmentación que clasifica el audio en voz, música, silencio… El segmentador nos permitirá dividir el archivo de audio entrante en segmentos correspondientes a cada hablante, que posteriormente serán transcritos con el reconocedor y almacenados en la base de datos como documentos independientes. Estos documentos contendrán, además de la transcripción, una serie de etiquetas (metadatos) que ayudarán a la búsqueda y localización del clip de audio dentro del archivo sonoro, como son tiempo de inicio (*begin*) y fin (*end*), el nombre del audio y el identificador del hablante (spk_id).

El análisis del audio puede mejorarse añadiendo más módulos al sistema, como sería el de identificador del hablante que nos proporcionaría el nombre del locutor. Pero también pueden colocarse antes del reconocedor otros extractores de características (entorno, estado de ánimo) para que, junto al segmentador y el reconocimiento del hablante, permitan un mejor funcionamiento de éste al usar modelos acústicos adaptados a cada locutor y entorno.

Para lograr un sistema de búsqueda efectivo, hay que conseguir extraer del audio la máxima información posible de modo fiable. Cuánto más precisión logremos en el segmentado, la transcripción, el reconocimiento del hablante…, mejores serán los resultados y más fácil encontrar el archivo deseado. Por ese motivo, el precio a pagar en el diseño de los módulos ha sido el tiempo de procesado. El trabajo previo de análisis de la voz tardará bastante pero no influirá en el tiempo de respuesta de la petición de usuario.

La labor principal ha sido adaptar los diversos módulos a la arquitectura empleada, haciendo uso de alguno de los desarrollados por el equipo de trabajo de la universidad, como el reconocedor. No va a realizarse un análisis comparativo entre las distintas técnicas de segmentación, p. ej., ya que el objetivo, más allá de ajustar los parámetros para su buen funcionamiento, es exclusivamente el montar el sistema y no estudiar todas las posibles técnicas del mercado.

1.2 Organización de la memoria

Se ha estructurado en seis partes. Tras la introducción que muestra la descripción general del proyecto y el contexto en que se realiza, explicaremos la arquitectura que rige todo el proyecto: su funcionamiento, estructura, componentes que la integran, ficheros de configuración, etc., dejando claro cómo opera el envío de paquetes (comunicación entre módulos), cuestión importante a la hora de adaptarlos a la arquitectura. En el punto 3 se describen todos los módulos del sistema por separado, indicando cuál es su función dentro del procesado para, en el apartado 4, pasar a relatar cómo se integran y se entrelazan en el sistema final. Ahí se detallará la interrelación que guardan y qué tipo de paquetes recibe cada uno de los servicios o módulos. Por último, contamos con un apartado de resultados, dónde ilustraremos lo conseguido en cada parte del proceso, y otro de conclusiones, dónde se indicará el posible trabajo futuro y las modificaciones para mejorar este sistema inicial, así como la valoración personal del trabajo.

El único anexo incorporado explica con detalle los componentes y funcionamiento de la arquitectura EDECAN, y se muestran los archivos de configuración que rigen nuestro sistema.

2 Arquitectura EDECAN

Antes de empezar a explicar cada módulo de la aplicación, comenzaremos describiendo el funcionamiento de esta arquitectura diseñada por la propia Universidad de Zaragoza en colaboración con otras universidades españolas para su expansión.

La interfaz de comunicaciones EDECAN [2] puede considerarse un *middleware* multiplataforma para el desarrollo de sistemas distribuidos sobre IP capaz de soportar cualquier tipo de servicios en los sistemas operativos Windows y Linux. Esta arquitectura permite la creación de nuevos módulos con nuevas funcionalidades a través del uso de un conjunto de librerías, protocolos y herramientas. El objetivo de su desarrollo ha sido facilitar la implementación de sistemas distribuidos a través de un proceso sencillo y en la medida de lo posible, independientemente de la plataforma hardware elegida o del sistema operativo empleado.

Siguiendo un esquema de comunicaciones “cliente-servidor”, aparece definido como “cliente” un gestor de comunicaciones (gestor.exe), que tiene la función de iniciar los servicios y actuar como enrutador de paquetes entre ellos. Por otra parte existe un súper-servidor (super_server.exe) que acepta las conexiones entrantes del gestor y lanza cada uno de los servicios en un nuevo proceso, dejando la responsabilidad de la conexión al propio servicio. Finalmente, aparece un servicio genérico (service.exe) sobre el que se puede implementar cualquier tipo de servicio capaz de enviar y recibir datos. Es decir, mientras super_server.exe gestiona la conexión inicial, service.exe envía los paquetes y los recibe directamente del gestor de comunicaciones.

Además, el gestor de comunicaciones deberá ser manejado con otro módulo del sistema llamado controlador (controlador_edecan.exe), el cual se conecta a él como un cliente y le envía comandos para realizar modificaciones sobre el sistema montado, como conectar servicios, desconectarlos, dar de alta nuevos servicios, o modificar la tabla de rutas del sistema. Este programa se conecta al gestor mediante *sockets* [5] (métodos de comunicación definidos por una dirección IP, un protocolo de transporte y un número de puerto que permiten intercambiar cualquier flujo de datos de manera fiable y ordenada entre procesos, ubicados o no en computadoras distintas) y se encarga de montar el sistema.

Los servicios disponibles, que consultará service.exe, aparecen en un fichero de configuración (services.xml). Además, cada servicio o módulo se define a partir de una librería dinámica (.dll o .so) y de un archivo de configuración que define su comportamiento frente a los diferentes eventos que puedan producirse. Esto permite montar un servicio sin utilizar librerías de comunicaciones propias del sistema operativo y del lenguaje de programación; para enviar/recibir datos únicamente hay que escribir/leer en una dirección de memoria. En el anexo se amplía la información sobre los componentes básicos de la arquitectura y cómo es el contenido de los archivos de configuración.

La topología de red está centralizada a partir del gestor de comunicaciones. Todos los paquetes que envíe algún servicio serán recibidos por el gestor y reencaminados hacia el servicio indicado. Por defecto, el gestor enrutará los paquetes hacia uno o varios destinos según la tabla de enrutamiento estática definida cuando el controlador web montó el sistema. Sin embargo, es posible definir una cabecera especial para un paquete, de manera que éste llevará un destino forzado hacia el servicio que indique dicha cabecera. Para eso el servicio que genere el paquete tiene que incluir los atributos 'from y to' en la raíz, así se enviará hacia el destino indicado por 'to'.

Hay definidos dos modos en el envío de paquetes: binario (seguía la tabla de rutas estática) y XML (tabla de rutas o destinos forzados). Estos dos tipos de paquetes se identificaban por el primer bit de su cabecera: 0 – paquete binario, 1 – paquete XML. El problema que presentaba esta configuración es que obligaba a mandar los envíos binarios a todos los servicios conectados e interesaba que, igual que pasaba con los paquetes de texto (XML), se pudiese diferenciar qué información generada por un servicio se enviaba a cada destino. Por eso se implementó la Conmutación de Paquetes en Modo Circuito Virtual para aumentar la eficiencia, estableciendo un circuito virtual para paquetes binarios.

El controlador es el encargado de mandar el paquete XML de establecimiento que indica el comando que se ejecutará en el servicio destino para los paquetes binarios. Transmitidos estos, se liberará el circuito virtual.

Para identificar los paquetes tendremos ahora 8 bits quedando la tabla de enrutamiento del gestor así: 0 broadcast, 1 XML, 2-255 binarios por circuito virtual.

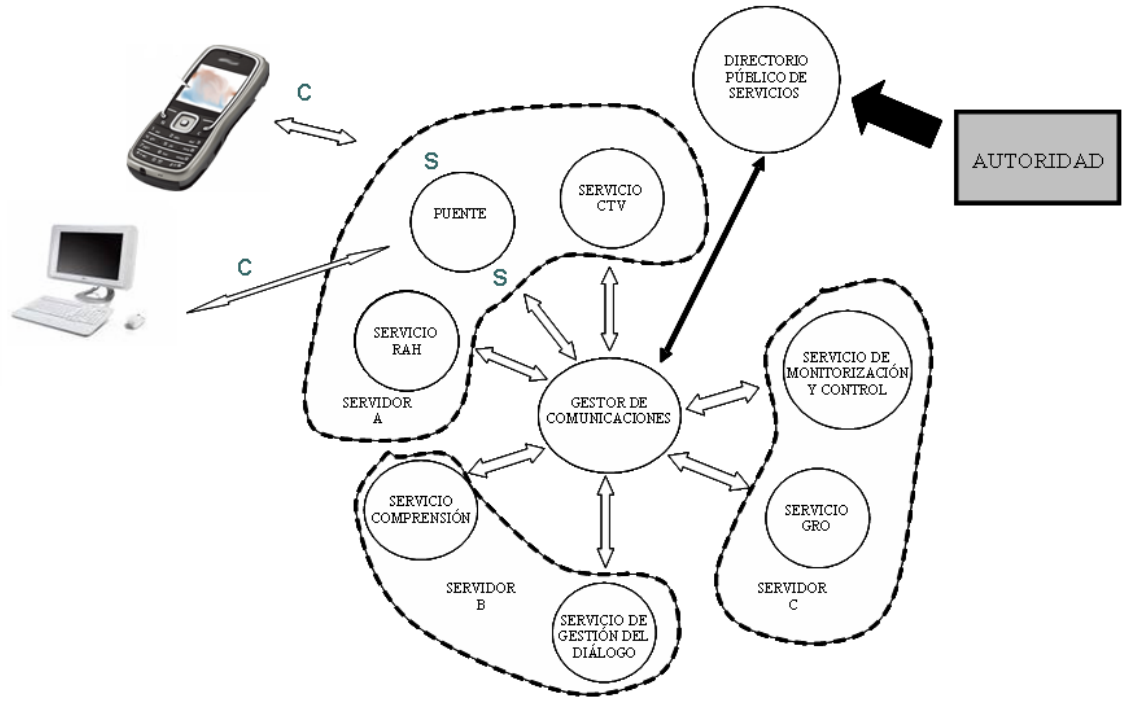

Figura 2.1: *Estructura* de un *sistema de diálogo hablado basado en la Arquitectura EDECAN*

3 Módulos

A continuación vamos a describir los principales módulos del sistema, que coinciden con los que integran la parte de ingesta del audio. Los módulos puente y servicio texto, que forman parte de la aplicación del buscador web, serán descritos en el siguiente apartado (sistema).

Son seis módulos: el de audio, que suministrará las muestras crudas del fichero; el parametrizador, que extrae las características necesarias para el resto de módulos; el detector de actividad vocal, que separa el ruido de fondo de la voz; el segmentador, que establece barreras entre los distintos hablantes; el reconocedor, que suministrará la transcripción del audio y, por último, la consola de control, a través de la cual el usuario podrá manejar el sistema indicando el nombre del archivo que desea procesar.

3.1 Audio

Este servicio genérico, ya implementado, se encarga de controlar (*play, stop, pause*) los archivos de audio, abrirlos, cerrarlos y extraer la información que necesitamos. Permite analizar distintos tipos y adaptarlos para el posterior procesado, extrayendo de los diferentes formatos (.wav, .mp3) las muestras crudas de audio (.raw) que va enviando, debido a un buffer de 4096 bytes que tiene y a la frecuencia de análisis (16 KHz) a la que remuestrea la señal, en paquetes de 25 tramas o *frames*.

El uso de este módulo da mayor versatilidad al sistema que si se hubiese integrado en el servicio consola de control. A la salida, tendremos las muestras crudas correspondientes a la frecuencia de muestreo tanto para audios de entrada monos como estéreos. Para esto es útil la información que, a través de los paquetes de control, el módulo audio envía al resto de módulos conectados según la tabla de rutas, antes del envío de los paquetes de audio. Según el número de canales procederemos a calcular los parámetros. El extractor de parámetros simplemente se queda con las muestras de un canal en caso de audio estéreo (análisis mono).

3.2 Parametrizador

El param_extractor es un módulo básico en el proceso de análisis de la voz. A él llegan en primer lugar las muestras de audio para calcular todos los parámetros necesarios en los módulos posteriores. Así se obtienen los coeficientes FFT, MFCC y las derivadas y derivadas segundas a partir de los *frames* de voz, analizando ventanas de 25 ms de longitud con un desplazamiento de 10 ms. La frecuencia de muestreo (fm) es configurable y se ha fijado en 16 KHz.

El parametrizador es pues un extractor de características que se encargará de enviar a cada módulo los parámetros que éste necesita para su correcto

funcionamiento. Así, en primer lugar calcula la transformada rápida de Fourier (FFT) útil para la detección de actividad vocal, enviándose 256 coeficientes por trama, ya que de los 512 que tiene la transformada, por simetría, la otra mitad es redundante.

Para la segmentación, la representación de la voz está basada en los MFCC (*Mel Frequency Cepstrum Coefficients* [6], [7]). Es una parametrización robusta consistente en coeficientes cepstrales en escala frecuencial Mel, siendo el Cepstrum la anti-transformada de Fourier del espectro de la señal en escala logarítmica. El que las bandas de frecuencia estén situadas logarítmicamente (según escala Mel), modela la respuesta auditiva humana mucho mejor que si las bandas estuvieran espaciadas linealmente como pasaba en la FTT. Esto permite un procesado de datos más eficiente.

La señal de voz, muestreada a 16 KHz, es segmentada en tramas (de 25 ms cada 10 ms como ya hemos visto) y cada trama es representada por un vector de características que contiene un coeficiente de energía (logarítmica) y 12 coeficientes cepstrales. Los 12 coeficientes cepstrales (MFCC) son los que mandamos al segmentador, el vector de características que necesitará el algoritmo de segmentación *Bayesian Information Criterion*.

Estos MFCC se utilizarán también para el reconocedor, cuyo vector de características se compone de 39 coeficientes: los 12 coeficientes MFCC más el término de energía logarítmica (13), junto a sus derivadas y derivadas segundas.

El parametrizador es el primer bloque de las funciones de detección vocal, segmentación y reconocimiento del habla que, en esta arquitectura por comodidad y versatilidad, se ha implementado externamente como módulo común compartido por todos ellos.

3.3 Módulo de detección de actividad vocal

El primer paso del proceso, una vez preparadas las muestras de audio con sus características calculadas, es el VAD (*Voice Activity Detection*). Este módulo identifica qué muestras son de voz y cuáles no, y supone un pre-procesado para el segmentador. Es importante para las siguientes etapas tener en cuenta solo la voz y no procesar ruido, por eso se pasa la salida del VAD consistente en una ristra de unos y ceros que indican qué muestra es voz (1) y cuál no (0). Las muestras con VAD a 0 no serán tenidas en cuenta el resto del análisis.

Para determinar los periodos de voz/no voz se emplea un test de hipótesis. La decisión se basa en un vector de observación llamado vector de características, el cual sirve como entrada a la regla de decisión que asigna un vector de muestras a una de las clases dadas.

La selección de un adecuado vector de características para la detección de la señal y una robusta regla de decisión, es un desafiante problema que afecta el desarrollo de VADs trabajando bajo condiciones ruidosas [8].

Para evitar que en condiciones muy ruidosas (SNR muy baja) se produzcan muchos errores, se utiliza un detector robusto, que presenta dos hipótesis de decisión:

$$H_0 : x=n \qquad\qquad H_1 : x=n+s$$

H_0 significa que la señal detectada es solo ruido, y H_1 que la señal se compone de ruido y voz. Por eso es preciso estimar las estadísticas del ruido obtenido.

Existen diferentes métodos VAD: umbrales de energía, detección del tono, análisis espectral, tasa de cruce por cero, medida de la periodicidad, estadísticas de orden superior o combinaciones de diferentes características. En nuestro caso emplearemos el análisis espectral, a través de la FFT, para identificar las muestras de voz/no voz.

El diagrama de bloques de un VAD es mostrado en la siguiente figura. Consiste en un proceso de extracción de características (visto en el apartado anterior), un módulo de decisión, y una etapa de alisado con la que se pretende pulir la decisión anterior (*Decision smoothing*).

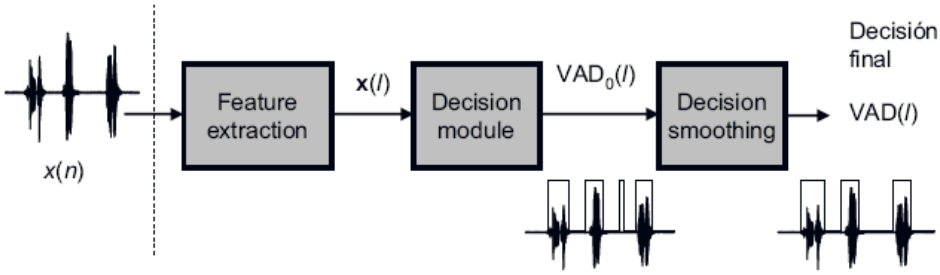

Figura 3.1: *Diagrama de bloques de un VAD*

3.3.1 Decision module

El módulo de decisión define la regla (LTSD [9], [10]) o método para asignar una clase al vector de características x (FFT). Para que el VAD funcione correctamente, hemos de tener el vector de características, por eso se almacenan primero los parámetros del audio antes de pasarlos por este decisor, ya que el parametrizador los calcula y envía por cada paquete de audio recibido (compuesto por unos 25 *frames*).

El método considera un test de dos hipótesis, donde la regla de decisión óptima que minimiza la probabilidad de error es el clasificador de Bayes. Dado un vector de observación para ser clasificado, el problema se reduce a seleccionar la clase H_0 o H_1 con mayor probabilidad a posteriori.

$$P(H_1 \mid \mathbf{x}) \underset{H_0}{\overset{H_1}{\gtrless}} P(H_0 \mid \mathbf{x})$$

La FFT de la voz limpia y la del ruido son asintóticamente variables gaussianas aleatorias independientes, por eso los diversos métodos se fundamentan en medidas de distancia. En concreto, el algoritmo de detección propuesto para este VAD robusto se basa en la estimación de los mayores términos de la envolvente espectral (LTSE) y la medida de la divergencia espectral (LTSD) entre la señal de voz y el ruido. Asume que la información más importante para detectar la actividad de voz sobre una señal ruidosa radica en la variación temporal de la magnitud de su espectro.

La señal de voz ruidosa *x*(*n*) es segmentada en tramas solapadas y X(k,l) es su amplitud espectral para la banda k-ésima en la trama l, definiendo LTSE de orden N:

$$\mathrm{LTSE_N}\ (\mathrm{k},\mathrm{l}) = \max\{\mathrm{X}(\mathrm{k},\mathrm{l}+\mathrm{j})\}_{\mathrm{j=-N}}^{\mathrm{j=+N}}$$

La regla de decisión VAD queda formulada mediante el LTSD (*Long-Term Spectral Divergence*) de orden N, definida como la desviación del LTSE (*Long-Term Spectral Envelope*) respecto al promedio de la magnitud espectral del ruido *N*(*k*) para la banda *k*, con *k*= 0, 1, …, *NFFT*-1.

$$LTSD_N\ (l) = 10\ log_{10}\left(\frac{1}{NFFT}\sum_{k=0}^{NFFT-1}\frac{LTSE_N{}^2\ (k,l)}{N^2\ (k)}\right)\ \begin{matrix}H_1\\ >\\ <\\ H_0\end{matrix}\ \eta$$

El umbral de decisión (*η*) es un parámetro fijado inicialmente que se va actualizando con las muestras recibidas. El VAD está diseñado para ser adaptativo a las variaciones temporales del ruido ambiental, por eso presenta un algoritmo para la actualización del espectro del ruido durante los periodos de no-voz.

3.3.2 Decision smoothing

Permite recuperar periodos de voz que han sido enmascarados por ruido acústico debido a la reducida energía de la señal en las palabras iniciales y finales. Así, las llamadas estrategias de *hang-over* extienden y suavizan la decisión VAD, retardando la clasificación. Estos algoritmos se desactivan para SNR altas, de este modo mejora también la detección de no voz en bajas condiciones de ruido.

3.4 Segmentador

El objetivo de la segmentación y clasificación del audio es la partición y el etiquetado del flujo de audio entrante en voz, música, anuncios, ruido de fondo ambiental u otras condiciones acústicas no homogéneas. Esta etapa preliminar es necesaria para recuperar información de audio como la transcripción de forma efectiva, ya que solo nos interesaría la voz.

Los segmentos de voz y no voz tienen diferente distribución de características tanto en el dominio del tiempo como en el de la frecuencia y en otros dominios (cepstral). Así que la clasificación basada en rasgos característicos es generalmente un método efectivo.

En este proyecto se va a utilizar la segmentación para detectar el cambio de hablante, ya que gracias al VAD hemos aislado los sonidos de fondo de lo que presenta mayor energía (principalmente voz, aunque también puede ser música o ruidos fuertes). El segmentador simplemente establece barreras entre fragmentos heterogéneos de audio.

El objetivo de una efectiva segmentación por hablante difiere de la meta de un reconocimiento automático de voz (ASR). Los procesos y modelos que son útiles para ASR, no necesariamente son apropiados para segmentación. Las características usadas para reconocimiento del habla intentan minimizar las diferencias entre los hablantes y las distintas acústicas ambientales, y maximizar las diferencias entre fonemas; mientras que en la segmentación por hablante es preferible maximizar los rasgos del locutor y minimizar la varianza de los fonemas simultáneamente para producir segmentos que contienen un único evento acústico o hablante.

Si los segmentos con un único hablante obtenidos son más largos que 5 s, el criterio de información bayesiana (BIC) y muchas aproximaciones basadas en la medida de distancias pueden hacer realizable el desarrollo de la segmentación. Sin embargo, estos métodos son insuficientes cuando tenemos segmentos más cortos (menores de 5 s). Se propone pues el uso de una nueva métrica de distancia, el T^2-mean [11], para tratar este problema, además de desarrollar una rutina de compensación de falsa alarma dentro del esquema de segmentación que se conoce como *clustering*. El algoritmo es un método de segmentación compuesto (CompSeg [12]) que se muestra en la figura 3.2.

Las estadísticas T^2 son usadas para preseleccionar los límites candidatos de segmentación, seguidas por el BIC para ejecutar la decisión de segmentación. El algoritmo propuesto también incorpora un esquema de ventana de tamaño variable que se va incrementando.

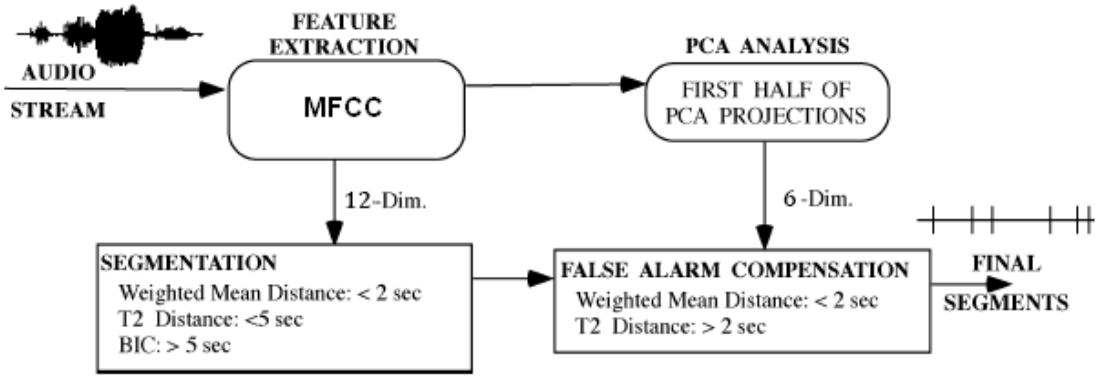

Figura 3.2: *Diagrama de bloques del algoritmo CompSeg*

El BIC (*Bayesian Information Criterion* [13]) supone que cada bloque de voz homogéneo acústicamente puede ser modelado por un proceso gaussiano multivariable $X \sim N(\mu, \Sigma)$, quedando la segmentación como un problema de selección de modelo entre los siguientes dos modelos anidados:

$$M_1: X = x_1, x_2, \ldots, x_N \sim N(\mu, \Sigma) \text{ y}$$
$$M_2: X = x_1, x_2, \ldots, x_b \sim N(\mu_1, \Sigma_1);$$
$$x_{b+1}, x_{b+2}, \ldots, x_N \sim N(\mu_2, \Sigma_2).$$

El primer modelo asume que todas las muestras son independientes e idénticamente distribuidas a una única gaussiana, mientras el segundo considera una frontera en la trama b, de forma que las b primeras muestras pertenecen a una gaussiana y las últimas N – b a otra. Denotamos con $X = x_i \in R^d$, i = 1, 2, …, N la secuencia de vectores cepstrales basados en tramas extraídos del audio.

La diferencia entre los dos modelos vistos puede ser calculada como función del punto de corte b: $\Delta BIC(b) = BIC(M_2) - BIC(M_1)$. Según la regla BIC, la segmentación del audio en dos partes en la trama b será aceptada si $\Delta BIC(b) > 0$. $BIC(M_{1,2})$ se calcula gracias a los valores de las medias, las variancias, el número de muestras, la dimensión del vector de características cepstral y un factor de penalización debido a la complejidad del modelo. De forma genérica puede verse que

$$BIC(M_i) = \log P(D_1, D_2, \ldots, D_N \mid M_i) - \frac{1}{2} d_i \log N$$

donde d_i es el número independiente de parámetros y $P(D_1, D_2, \ldots, D_N \mid M_i)$ es la probabilidad de datos maximizada para el modelo dado. El término (1/2) d_i log N es substraído de la probabilidad logarítmica para penalizar por la complejidad del modelo. Esta penalización es necesaria porque el modelo M_1 (un único hablante) va a ir siempre con detrimento respecto a M_2 (dos hablantes) debido al menor número de parámetros implicados en la estimación (el segundo modelo tiene doble de parámetros al suponer dos gaussianas).

Para flujos de audio que contienen múltiples puntos de segmentación (varios hablantes), se aplica un algoritmo de detección secuencial en una ventana móvil que barre todo el flujo de audio. La ventana comienza desde el inicio del audio con un tamaño de 1s. Dentro de la actual ventana, el test BIC visto ($\Delta BIC(b)$) es evaluado para cada 1 < b < N para determinar si existe un cambio. La ventana se dobla si ninguna frontera es encontrada, o una nueva de 1s comienza desde la división detectada como la siguiente ventana.

El modelo gaussiano es ampliamente usado para modelar observaciones con función de densidad de probabilidad desconocida. La idea de usar estadísticas T^2 asume que las covariancias de $N(\mu_1, \Sigma_1)$ y $N(\mu_2, \Sigma_2)$ son iguales pero desconocidas, de forma que la única diferencia entre ellas es la media. Bajo esta presunción, se pueden usar más datos para estimar la covarianza y reducir el efecto de borde que se produce con fragmentos de 5s o menos debido a la insuficiencia de datos para la estimación solo con BIC. Aunque para ventanas menores de 2s, incluso la covariancia global es insuficiente, por eso se usa la distancia media ponderada.

El módulo segmentador del sistema EDECAN se compone de dos funciones principales: *segment* y *cluster*. *Segment* realiza el proceso de segmentación ya

explicado y *cluster* se encargará de agrupar los fragmentos pertenecientes al mismo hablante p.ej. en una entrevista. Esta agrupación consistirá únicamente en una compensación de falsa alarma, es decir, que si dos nodos fusionables son segmentos adyacentes en la rutina de agrupación del audio, ha habido un error en segmentación que puede ser corregido.

Conceptualmente, la compensación de falsa alarma es similar a la clasificación. Aquí, la distancia entre dos segmentos adyacentes es calculada y si la distancia está por debajo de un umbral, entonces ambos pertenecen a la misma clase (un falso punto de corte es encontrado), de lo contrario pertenecen a clases diferentes. Al igual que la segmentación, se aplica la métrica de distancias medias ponderadas mediante pesos para segmentos cortos y el T^2 para segmentos más largos.

3.5 Reconocedor automático del habla

El último paso para tener lista la información del audio en un archivo es obtener su transcripción gracias a un Reconocedor Automático del Habla (RAH [14]), de forma que tendremos el texto de cada segmento anterior. La complejidad de este módulo es elevada ya que hay varios factores a tener en cuenta para llevar a cabo esta tarea: acústica, fonética, fonológica, léxica y sintáctica. La dificultad está en hacer cooperar todas estas áreas de conocimiento para la obtención de la correcta transcripción. Debe ser capaz de reconocer las unidades acústicas de entre un conjunto de observaciones acústicas recibidas, y combinarlas para formar las palabras pronunciadas.

3.5.1 Dificultades

Existen muchos factores que influyen en la dificultad del proceso de RAH y por tanto en su rendimiento, pero entre todos ellos destaca la variabilidad. La variabilidad de la señal de voz depende tanto de factores intrínsecos al fenómeno de producción de voz como a factores externos al mismo. Dentro de los factores intrínsecos destacan los siguientes: acentos o formas de hablar de cada persona, distintas velocidades de producción, coarticulación, inclusión de ruidos (apertura y cierre de labios, respiración, sonidos de duda, p.ej., eh, uuh), condiciones acústicas, contexto de la conversación, estado anímico, etc. Entre los factores externos destacan: variabilidad en la cadena de conversión y transmisión de la señal eléctrica, debido a las diferencias entre las características de los micrófonos, líneas telefónicas, etc. y variabilidad en el ruido captado con la señal de voz, debido a la existencia en las proximidades del micrófono de otras fuentes sonoras.

A estos factores de variabilidad acústica habrá que añadir otros factores de variabilidad lingüística relacionados con las distintas formas dialécticas de un idioma, la utilización de palabras no contempladas en el vocabulario de la aplicación, la construcción de frases no permitidas por la gramática del lenguaje, la utilización de abreviaturas, los escenarios semánticos de las palabras, etc. Todo ello hace que el

reconocimiento automático del habla por parte de una máquina no sea un problema trivial.

3.5.2 Principios básicos

Básicamente, el reconocimiento del habla es un proceso de clasificación de patrones, cuyo objetivo es clasificar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos diccionarios de modelos acústicos y de lenguaje. Este proceso de clasificación supone, en primer lugar que la señal de voz puede ser analizada en segmentos de corta duración y representar cada uno de los segmentos mediante su contenido frecuencial de forma análoga al funcionamiento del oído, en segundo lugar que mediante un proceso de clasificación podemos asignar a cada segmento o conjuntos consecutivos de segmentos una unidad con significado lingüístico y finalmente, en tercer lugar, que mediante un procesador lingüístico podemos dar significado a las secuencias de unidades. Este último paso del sistema supone incorporar al sistema de RAH conocimiento acerca de la estructura acústica, léxica y sintáctica del lenguaje.

El funcionamiento del reconocedor se basa en dos módulos: modelo acústico y modelo de lenguaje. Consta de un vocabulario de decenas de miles de palabras apropiadas para un reconocedor de radio como es el caso (para otras aplicaciones como cine el listado de palabras es menor) y solo podrá procesar el léxico que se haya definido previamente junto a su transcripción fonética. El modelo acústico se basa en tri-fonemas; es un modelo fonético que tiene en cuenta el fonema vecino de la izquierda y de la derecha. Si dos fonemas tienen la misma identidad pero diferente contexto del de la izquierda y/o del de la derecha, entonces, son considerados tri-fonemas distintos. Los modelos basados en tri-fonemas son potentes porque capturan los efectos más importantes de la coarticulación, y son generalmente más consistentes que los modelos fonéticos independientes del contexto.

El conocimiento de las propiedades acústicas de los sonidos permite determinar qué fonema forma parte de la cadena de entrada y en un golpe determinado haremos una extracción de parámetros que nos segmentará la cadena sonora en diferentes fonemas que serán transcritos posteriormente. En los sistemas que se basan en modelos de fonética acústica, podemos distinguir tres etapas que analizan la entrada sonora y las frecuencias de los formantes que pueden variar según el trato vocal, velocidad de elocución, etc. Tras identificar los sonidos que pertenecen a la cadena de entrada, se realiza el reconocimiento lingüístico (identificación de palabras y frases). Para llevarlo a cabo, el sistema incorpora un diccionario de palabras posibles y un modelo de lenguaje que codifica la frecuencia de aparición de las diferentes palabras y su relación.

3.5.3 Arquitectura de un sistema RAH

Matemáticamente, el problema del reconocimiento automático del habla se puede formular desde un punto de vista estadístico. Para ello supongamos que **O** representa una secuencia de T medidas de la señal de voz (datos acústicos) y **W** es una secuencia de N palabras que pertenecen a un vocabulario conocido. La probabilidad

condicional P(**W**|**O**) es la probabilidad de que la secuencia de palabras **W** se haya pronunciado dada la observación de los datos acústicos **O**. El sistema de reconocimiento debe decidir en favor de la secuencia de palabras **W** que maximice la probabilidad P(**W**|**O**):

$$W = \underset{W}{\operatorname{argmax}}\, P(\mathbf{W}|\mathbf{O})$$

Utilizando la fórmula de Bayes podemos reescribir la probabilidad condicionada

$$P(W \mid O) = \frac{P(O \mid W)P(W)}{P(O)}$$

donde:

P(**W**) es la probabilidad de la secuencia de palabras **W.**

P(**O**|**W**) es la probabilidad de observar la secuencia de datos acústicos **O** cuando se pronuncia la secuencia de palabras **W.**

P(**O**) es la probabilidad de la secuencia de datos acústicos **O.**

Sin embargo, como la probabilidad de la secuencia de datos acústicos P(**O**) es la misma independientemente de la secuencia de palabras pronunciada, ésta puede ser eliminada en el proceso de maximización (la secuencia de palabras que da el máximo no varía). De esta forma obtenemos la fórmula fundamental del reconocimiento automático del habla:

$$\boxed{W = \underset{W}{\arg max} P(O|W)P(W)}$$

La secuencia de palabras reconocida es aquella que maximiza el producto de dos probabilidades, una P(**O**|**W**) que relaciona los datos acústicos con la secuencia de palabras y que denominaremos ***modelo acústico*** y P(**W**) que únicamente depende de la secuencia de palabras y que denominaremos ***modelo de lenguaje***.

El modelo acústico se basa en los modelos ocultos de Markov. Un Modelo Oculto de Markov (HMM: *Hidden Markov Model*) es un proceso doblemente estocástico que encierra un proceso estocástico fundamental que no es observable (es oculto), pero se puede descubrir a través de otro conjunto de procesos estocásticos que producen la secuencia de símbolos observados.

La figura 3.3 muestra los bloques básicos de un sistema de reconocimiento automático del habla basado en la anterior fórmula. En la figura se distinguen dos procesos diferenciados:

1. **Entrenamiento:** Fase en la que el sistema aprende, a partir de muestras de voz y texto, los modelos acústicos P(**O**|**W**) y los modelos de lenguaje P(**W**).
2. **Reconocimiento:** Fase propiamente dicha de reconocimiento automático del habla en la que la señal acústica es transcrita en una secuencia de palabras de acuerdo con la fórmula fundamental del RAH.

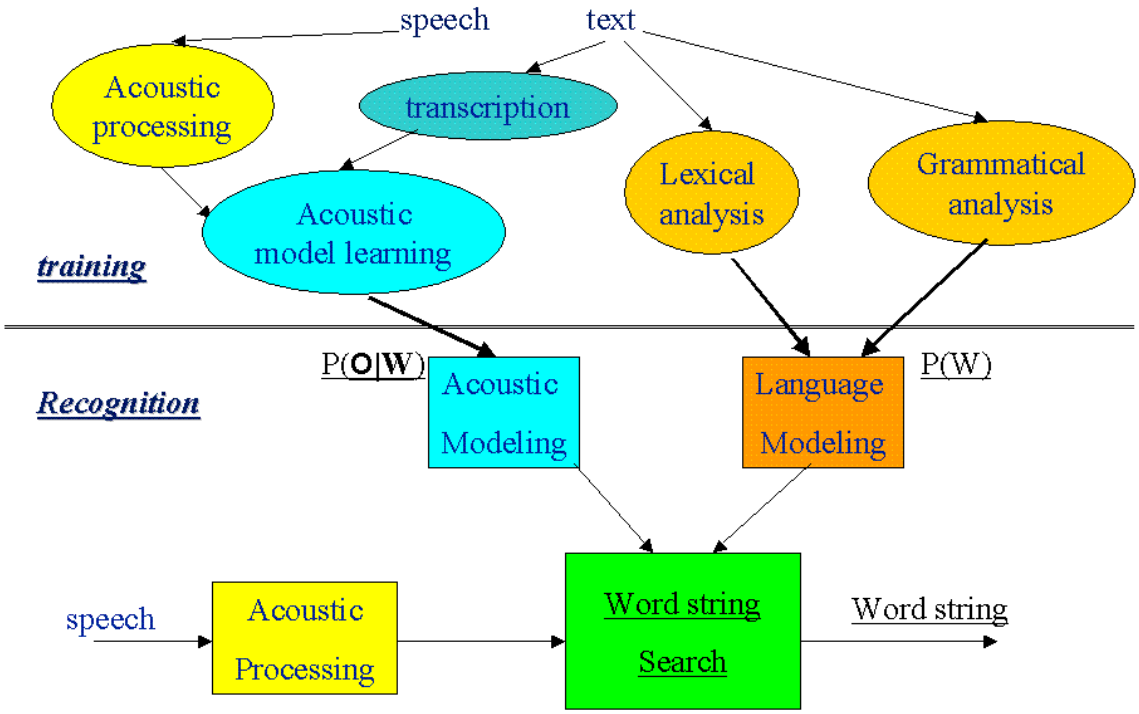

Figura 3.3: *Bloques básicos de un sistema de reconocimiento automático del habla*

Esta segunda fase es la que tenemos implementada en el proyecto, haciendo uso de los modelos ya aprendidos. El compromiso que se establece es calidad frente a tiempo de procesado. Como éste es offline, lo importante es obtener la mejor transcripción posible.

3.6 Consola de control

Es el módulo inicial del sistema, el que interactúa con el usuario gracias a un hilo, creado en dicho servicio, que permanece esperando los datos de entrada (comando *audio exec* con el nombre del fichero, el protocolo y la frecuencia de muestreo que se van a usar). Desde ahí lanzaremos el resto de módulos (envío de paquetes de control indicando que pueden empezar a ejecutarse) y presentaremos los resultados.

Este interfaz en modo consola guarda el .vad y .raw del archivo de audio y muestra por pantalla el texto transcrito. Una mejora sería usar un interfaz gráfico (GUI) que hiciese más atractiva la aplicación.

4 Sistema

El sistema de recuperación de información completo se estructura en dos partes. Cada una corresponde a un subsistema montado sobre la arquitectura EDECAN que se relacionan entre sí de modo que la salida del primero es la entrada del segundo. Estas partes son la ingesta del audio y el buscador web.

La ingesta del audio es el núcleo o parte principal del proyecto. Los diversos módulos que lo componen hacen paso a paso todo el procesado que necesita el audio para su etiquetado y posterior ingesta en la base de datos, obteniéndose así, en un mismo documento de texto, las transcripciones de todos los audios junto a otros datos relevantes para su recuperación: tiempo de inicio y fin, identificador del hablante, nombre del audio. Finalizada esta etapa, el fichero de salida es subido a la Base de Datos (BD), lo que hace que la ingesta de los documentos al archivo sea offline.

La parte del buscador web accede a esa información generada en la fase anterior mediante un motor de búsqueda. Es lo que hemos llamado *Query & Retrieval* (consulta y recuperación) en la figura.

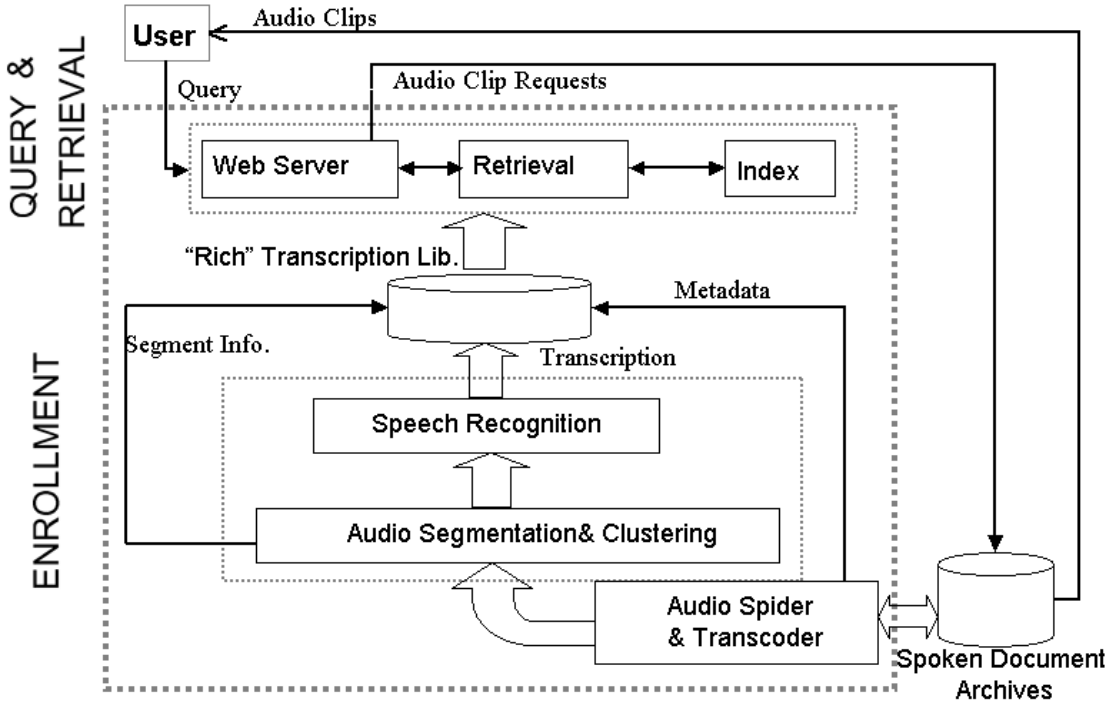

Figura 4.1: *Visión general de la arquitectura de un sistema de búsqueda de voz*

4.1 La ingesta del audio

El gestor de comunicaciones es el nodo central (*hub*) de la aplicación y está controlado mediante línea de comandos por un servicio de circuito virtual que permite establecer

el envío de paquetes binarios desde el parametrizador hasta el VAD, el segmentador y el reconocedor a través de los circuitos virtuales creados. Por el circuito virtual 2 se envían las FFT al VAD, por el 3 los MFCC al BIC y por último, el circuito 4 permite el envío de MFCC+Δ+ΔΔ al RAH.

La consola de control permanece a la espera de que el usuario introduzca el comando con el audio que desea analizar y, una vez enviado, inicializa los módulos audio y param_extractor. Dichos comandos son empaquetados en XML y llevan destino forzado, con lo cual llegarían al módulo adecuado aún cuando no apareciesen conectados al servicio consola en la tabla de rutas estática. Aquí hacemos uso de un encaminamiento dinámico, ya que diferentes comandos despiertan los distintos servicios de la arquitectura.

El servicio audio comenzará abriendo el fichero y enviando las muestras crudas tanto a la consola de control (donde se escriben en un archivo) como al parametrizador, que irá calculando todos los parámetros paquete a paquete. Todos sus envíos son *broadcast*.

Cuando ya tenemos todo el audio leído y sus parámetros calculados, es a través de la consola como se indica al VAD que todo el audio ha sido analizado y que puede empezar a calcular los unos y ceros según detecte voz o no. El resultado del VAD es pasado al segmentador, que lo usará como entrada junto a los coeficientes MFCC de las tramas para obtener las fronteras del audio y el número de hablantes.

Los resultados del VAD y el RAH son empaquetados hacia la consola de control para ser representados o guardados en su correspondiente fichero.

A este sistema podría incorporarse un módulo adicional tras el segmentador que se encargase de identificar al locutor (spk_id) de tener un registro de voces. Este reconocimiento del hablante permitiría tener un nombre en el identificador en vez del número obtenido con el segmentador.

Un sistema de reconocimiento automático del locutor permite al SDR identificar a la persona que ha emitido la señal. Para ello hay que entrenar al sistema introduciendo muestras de voz para que éste pueda crear una serie de patrones. Hecho esto, el sistema estará listo para reconocerlo y el RAH no podrá empezar a funcionar hasta que tenga la salida de ambos módulos (segmentador e identificador de hablante).

La figura 4.2 muestra todos los caminos lógicos establecidos en el sistema, sin olvidar que todo paquete pasa de un servicio a otro a través del gestor de comunicaciones.

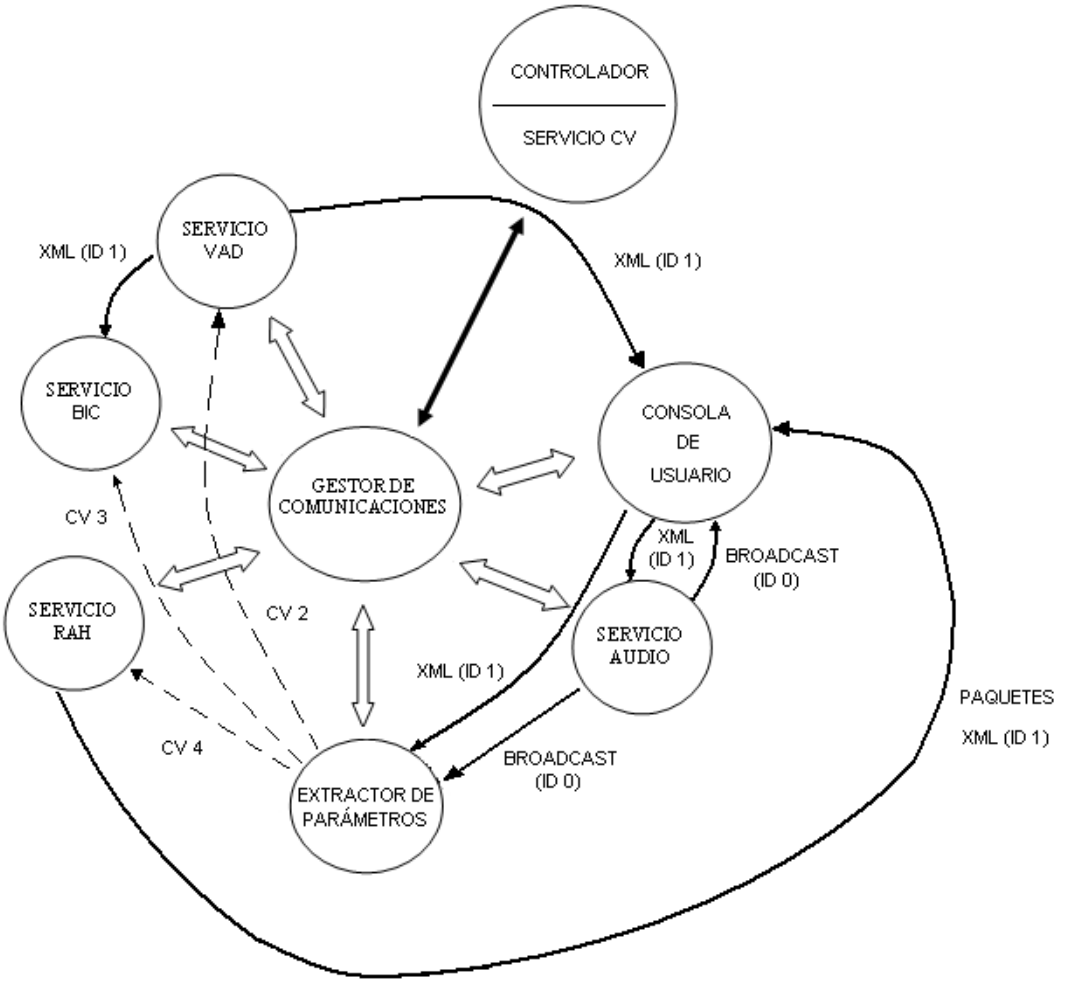

Figura 4.2: *Esquema del sistema Módulo análisis voz*

4.2 El buscador web

La interacción con el usuario se lleva a cabo a través de páginas web. Buscador.html solicita al usuario que introduzca la descripción del audio que desea escuchar. Es lo que llamamos *query*. Dicha *query* es recogida por Resultados.php que se encargará mediante un servlet de acceder a nuestro sistema EDECAN montado sobre el clúster de voz. Este sistema se conoce como webservice_texto y se compone de dos módulos: el puente y el servicio texto. El puente hace de intermediario en la conexión entre las dos máquinas y se conecta como servidor tanto con el gestor de comunicaciones como con el servlet lanzado por la web. El servicio texto, por su parte, se encarga de procesar la *query* y llamar al MG para que devuelva el texto correspondiente a la petición (transcripciones y etiquetas). La respuesta del MG será empaquetada por el servicio texto y enviada al puente a través del cual llegará como variable a Resultados.php que se encargará de filtrar las etiquetas para representarla adecuadamente.

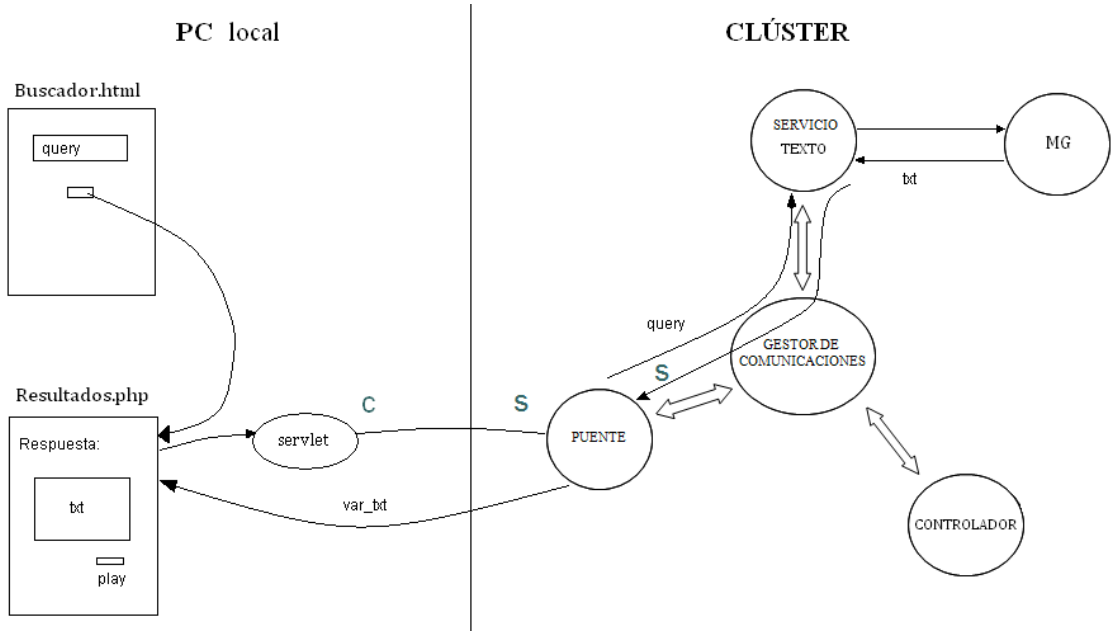

Figura 4.3: *Esquema del sistema webservice*

A continuación, se explica con detalle el funcionamiento de cada uno de los elementos que lo integran.

4.2.1 Motor de búsqueda

Esta herramienta permite la compresión e indexación de los textos a los que se accede con cada petición de usuario gracias a un comando que construye la Base de Datos a partir del archivo XML de salida de la ingesta automática del audio, donde cada párrafo corresponde a la transcripción con sus etiquetas de un segmento de audio detectado por el BIC. Este archivo queda subdividido en párrafos, de forma que cada uno es un documento y es lo que se visualizará como resultado.

El motor utilizado ha sido el MG (*Managing Gigabytes* [1]), un software libre disponible en la red que trabaja bajo el sistema operativo Unix. Esta potente herramienta tiene en cuenta solo las raíces gramaticales de las palabras (lexemas) a la hora de realizar la búsqueda. Es lo que se conoce como *stemming* [15]. Además elimina las palabras vacías de las consultas (*stop words*), proceso que descarta los términos carentes de contenido semántico discriminativo (artículos, preposiciones, conjunciones, adverbios, demostrativos, posesivos, algunos pronombres, verbos auxiliares (ser, estar, haber) y verbos muletilla como tener) y reduce drásticamente el vocabulario de la colección.

El reducir cada término restante a su raíz morfológica, disminuye aún más el vocabulario. Palabras con el mismo lexema (unidad mínima con significado), aunque pudiendo contener morfemas (sufijos o prefijos – ud. mín. de análisis gramatical), se representan con idéntica raíz morfológica.

El *stemmer* es pues un analizador morfológico que agrupa palabras relacionadas bajo un mismo concepto y usa una lista de *stopwords*. El utilizado por MG era Lovins

C, un conocido *stemmer* en inglés que ha sido reemplazado por la versión española de Snowball [16].

El archivo de configuración de MG permite establecer cómo representar la respuesta del buscador: texto del documento completo o solo cabeceras (primeros 50 carácteres), número máximo de documentos a mostrar (fijado a 10), etc.; y el tipo de *query* entrante, en nuestro caso *ranked* (lista de palabras).

El funcionamiento del motor de búsqueda es sencillo. Con la petición de usuario se accede a los documentos creados al montar la base de datos y busca similitudes (cantidad de veces que aparece cada una de las palabras que figuran en la *query*, sin tener en cuenta las *stop words* y fijándonos solo en la raíz, con lo cual buscamos textos que contengan palabras de la misma familia léxica). MG calcula así un factor de similitud y se representarán en orden, de mayor a menor relevancia, tantos documentos como se haya fijado o que presenten un factor de similitud no nulo (al menos aparece una vez alguna de las palabras escritas por el usuario).

4.2.2 Interfaz web

Es la aplicación que utilizará el usuario, lo único que éste verá de todo el sistema SDR montado. Se trata de una ventana de búsqueda, con un espacio para que el usuario escriba la palabra o palabras clave del audio que desea encontrar. Tras dar al botón buscar se visualizará una página de resultados, con los distintos documentos que hacen referencia a esa petición del usuario. Aparecerá la transcripción del fragmento de audio al que está vinculado y se podrá escuchar haciendo clic en el botón *play*. Además se incluye el identificador del hablante, que estará linkado a la URL de su página web, en caso de tenerla, o al buscador web por excelencia con los parámetros de búsqueda indicados (nombre del locutor). De esta forma podemos acceder cómodamente a la información web que haya acerca de los hablantes del archivo sonoro.

Este interfaz está montado sobre una aplicación Cliente-Servidor. Aquí se opera bajo Apache Tomcat: Apache es el servidor web y Tomcat el contenedor de servlets. Un servlet [17] es un programa que corre en el servidor y permite la creación de páginas dinámicas a partir de los parámetros de la petición que envíe el navegador web. Su ciclo de vida es el siguiente:

1. El cliente (usuario) solicita una petición al servidor vía URL.
2. El servidor recibe la petición. Si es la primera se utiliza el motor de Servlets para cargarlo. Si ya está iniciado, toda petición se convierte en un nuevo hilo.
3. Se llama al método service() para procesar la petición devolviendo el resultado al cliente.
4. El motor de un Servlet se apaga y se liberan los recursos abiertos.

Para el desarrollo de las páginas web, se han utilizado los siguientes lenguajes de programación: HTML, PHP y SMIL. HTML es el lenguaje de marcado predominante para la elaboración de páginas web y es usado para describir la estructura y el contenido en forma de texto (atributos, líneas en blanco, etc.). PHP nos permite la

creación de páginas dinámicas, como es el caso de la web de resultados. Además de ejecutar programas externos (lanza el servlet), permite trabajar con funciones y variables, p.ej., se hace uso de distintas funciones *string* para el filtrado de las etiquetas que aparecen en la respuesta y su correcta representación. SMIL es el acrónimo de *Synchronized Multimedia Integration Language* (lenguaje de integración multimedia sincronizada) y es un estándar del *World Wide Web Consortium* (W3C) para presentaciones multimedia. Permite integrar audio, vídeo, imágenes, texto o cualquier otro contenido multimedia y, gracias a él, podemos especificar el fragmento a reproducir (tiempo de inicio y fin), ya que partimos de audios de 15 min de duración.

4.2.3 Servicio web

Montado sobre la arquitectura EDECAN en el clúster de computación, donde reside el motor de búsqueda y la base de datos, este sistema suministra la respuesta requerida por la página web. La base de datos y el servidor web están alojados en terminales diferentes, por eso es necesario una herramienta que enlace y permita trasladar la información de un lado a otro. Esto se consigue mediante un servlet. Este programa crea un hilo de conexión con la máquina de voz (IP y puerto especificados en la definición del servlet) y cada vez que llega una consulta de usuario inicia un servicio: el puente_texto de la arquitectura.

Tras lanzar el gestor de comunicaciones el puente_texto, éste inicializa los *sockets* que le permiten comunicarse tanto con el servidor como con el servicio y crea un hilo (únicamente se aceptará una conexión por puente) que espera la llegada de una conexión entrante o *query.* Mientras dure esta conexión, recibirá los paquetes procedentes del servidor web que enviará a través del gestor al servicio principal: el ejemplo_texto. Una vez enviada la petición al otro módulo del sistema, esperará la respuesta de la máquina que recogerá posteriormente el servlet. El puente es necesario porque permite, sin modificar la estructura y la filosofía EDECAN, la conexión con el servidor web, que actúa como cliente.

El ejemplo_texto escribe la *query* de usuario en el fichero de entrada del MG, ejecuta el programa y recoge en una variable el contenido del fichero de salida, la respuesta que se visualizará en la página de resultados.

5 Resultados

A continuación mostraremos a título ilustrativo, no exhaustivo, el funcionamiento paso a paso de cada uno de los módulos de la arquitectura, la salida de los principales bloques que componen el sistema de análisis de la voz (procesado del audio). Después se visualizará la parte interactiva (buscador y web de resultados) con la que el usuario puede solicitar el archivo de audio que desee encontrar.

5.1 Funcionamiento del VAD y el segmentador

Ya hemos visto que el primer paso del análisis consiste en clasificar los fragmentos de audio como voz o no voz, y esta clasificación se completa a continuación con el segmentador, que establece fronteras entre los distintos hablantes.

Los resultados del VAD y de la segmentación obtenidos para 8 y 16 KHz (frecuencias de estudio en el desarrollo del proyecto) con un fragmento de audio extraído del Archivo Sonoro de Aragón Radio, que hace referencia a la preocupación por las obras de la expo una semana antes de la inauguración debido a la crecida del Ebro (**muestra3.wav**), son los siguientes:

Con 8 KHz:

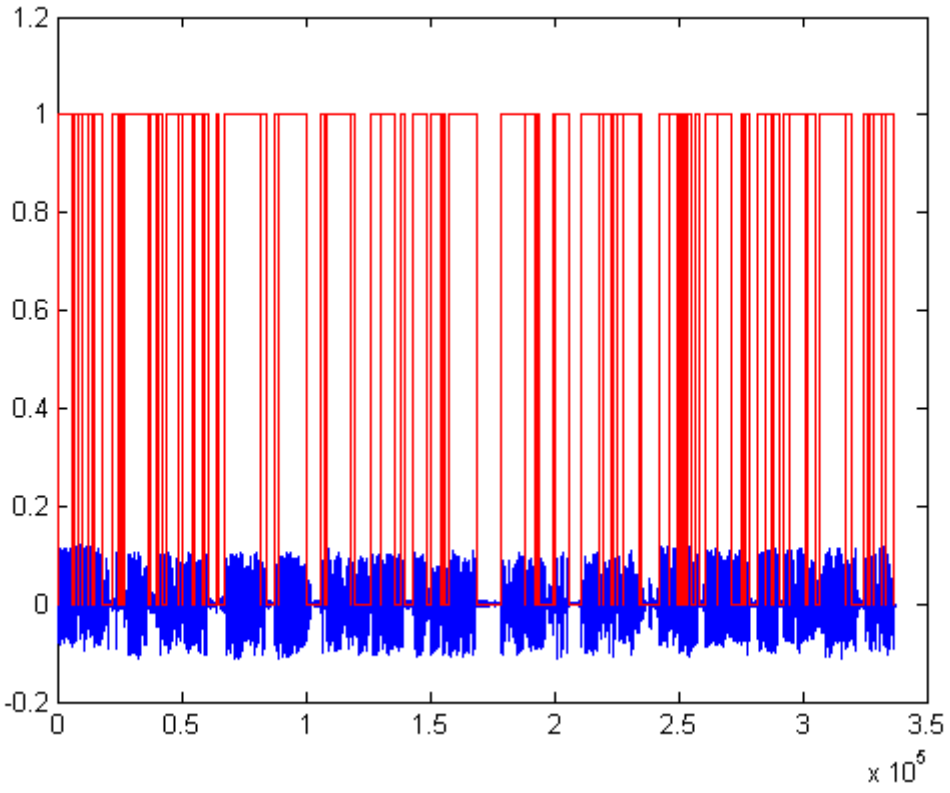

Figura 5.1: *Detección de voz de una señal de audio*

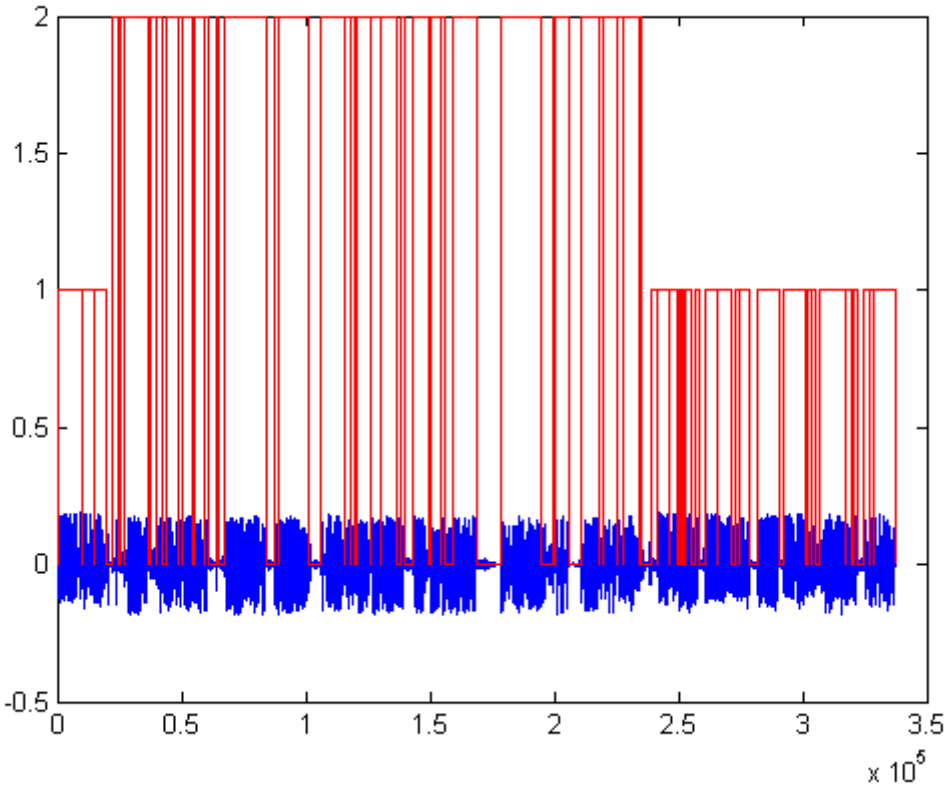

Figura 5.2: *Segmentación del mismo fragmento de audio*

Breaks: 250 2940
Spkids: 1 2 1

Los *breaks* (puntos de rotura o cambio) indican numéricamente el *frame* en que se pasa de un hablante a otro. En este ejemplo, se corresponde bastante con la realidad, donde primero habla el locutor, luego el invitado y de nuevo el locutor. Los puntos exactos de *break* son: 260, 3000.

Con 16 KHz:

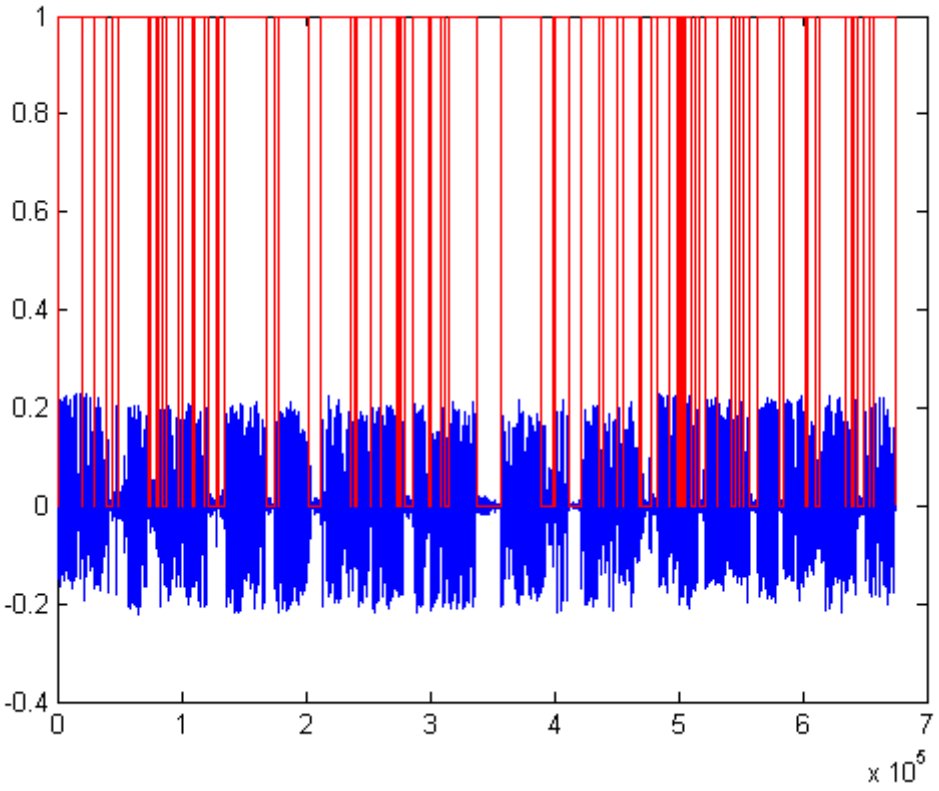

Figura 5.3: *VAD para fm= 16 KHz*

16 KHz es la frecuencia de operación del RAH, ya que fue entrenado para esa frecuencia de muestreo (fm). Es por tanto la frecuencia final que nos va a interesar. Aquí comparamos ambas fm para ver cuál es el precio a pagar en estos primeros módulos por usar un RAH entrenado a 16 KHz.

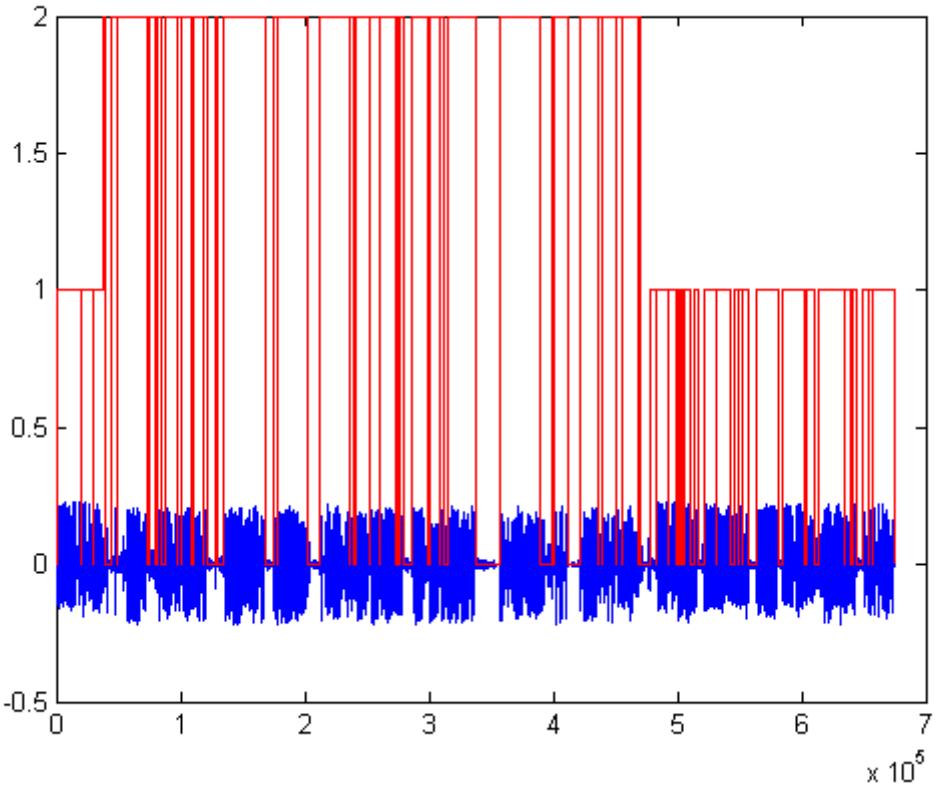

Figura 5.4: *Segmentación para fm=16KHz*

Con 16 KHz marca a priori más *breaks* que se corrigen con el *clustering*, algo que no pasa con 8 KHz pues a esta frecuencia VAD y segmentador son más exactos. No obstante, al final nos quedamos con los *breaks* adecuados y el número preciso de hablantes.

Breaks: 240 2940
Spkids: 1 2 1

Respecto al VAD, no se aprecia diferencia en este ejemplo, aunque de haberlas suelen ser pequeñas.

Para el audio **accidente ferroviario**, del que transcribiremos a continuación un fragmento, obtenemos el siguiente segmentado (el VAD es igual pero marca siempre 1 donde hay voz mientras la segmentación indica 1, 2, 3… dependiendo de quién habla):

Breaks: 940 2840 4410 4950 5830 6360 6850
Spkids: 1 2 3 4 5 6 4 3

Los breaks reales corresponden a los *frames* 970, 2830, 4400, 4950, 5830, 6350 y 6865. Mientras los hablantes eran: primero el locutor, luego la ministra de fomento, otra vez el locutor, y a continuación los testimonios de dos mujeres, un hombre y otra mujer, para finalizar con el locutor de nuevo. La mujer 3 y la mujer 1, aunque posiblemente sean personas distintas, es cierto que se parecen bastante en la voz, por

eso es normal que el segmentador se equivoque al identificarlas y les asigne el mismo identificador (4).

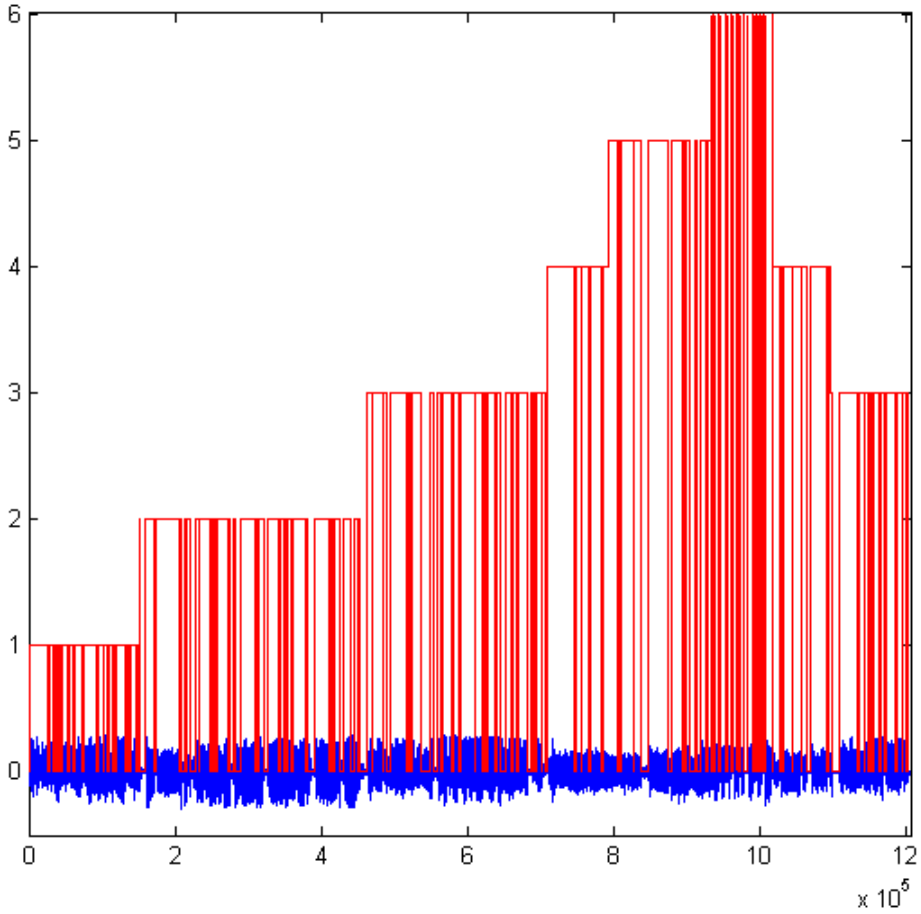

Figura 5.5: *Segmentado de accidente_ferroviario.wav (fm=16 KHz)*

El que la frecuencia de trabajo sea finalmente 16 KHz y no 8 KHz empeora ligeramente estos resultados, ya que pueden salir *breaks* de más o no detectar algunos. O como pasa en este caso, los *breaks* son correctos pero los spkids difieren ligeramente del óptimo, pues tras la ministra de fomento (spkid 2), vuelve a hablar el locutor y lo confunde con un nuevo hablante.

Con 1 denominamos siempre al primer locutor que encontramos, con 2 al siguiente y así sucesivamente. Si algún locutor se repite, volverá a marcarlo con el mismo identificador.

El resultado no es igual si partimos del audio original y se remuestrea a 16 KHz con el módulo audio de nuestro sistema, que si desde el wavesurfer ya lo transformamos a 16 KHz mono para mayor rapidez en el análisis. Es mejor no transformarlo y procesar directamente el audio original, como se haría en realidad.

5.2 Funcionamiento del reconocedor

De momento, para que el reconocedor funcione adecuadamente, solo es posible analizar fragmentos muy pequeños, ya que la complejidad aumenta con el mayor tamaño de los audios a tratar. Por ese motivo, hemos seleccionado un fragmento (muestra4.wav) del archivo accidente_ferroviario.wav, correspondiente a la primera frase que dice el locutor, para ver su transcripción.

Primero hemos comprobado que al estar entrenado con fs=16KHz, el reconocedor no funciona para nada con 8K, no es capaz de capturar las palabras a esta frecuencia. Solo se reconoce howard:
<recognized>
<text> howard </text>
<word time='4.55'> howard </word>
</recognized>

Sin embargo a 16KHz, aunque no tenemos reconocimiento perfecto, si que más o menos captura todas las palabras pronunciadas:

```
RESULTS: <recognized>
        <text> se espera una gran sima de cuarenta y dos horas contando los deta
lles del siniestro y que puedan estar ma~nana en el operativo del servicio ferro
viario magdalena alvarez ministra de fomento </text>
        <word time='0.13'> se </word>
        <word time='0.36'> espera </word>
        <word time='0.49'> una </word>
        <word time='0.81'> gran </word>
        <word time='1.09'> sima </word>
        <word time='1.18'> de </word>
        <word time='1.60'> cuarenta </word>
        <word time='1.64'> y </word>
        <word time='1.75'> dos </word>
        <word time='1.94'> horas </word>
        <word time='2.59'> contando </word>
        <word time='2.78'> los </word>
        <word time='3.20'> detalles </word>
        <word time='3.33'> del </word>
        <word time='3.76'> siniestro </word>
        <word time='3.81'> y </word>
        <word time='3.88'> que </word>
        <word time='4.31'> puedan </word>
        <word time='4.55'> estar </word>
        <word time='5.10'> ma~nana </word>
        <word time='5.22'> en </word>
        <word time='5.40'> el </word>
        <word time='5.95'> operativo </word>
        <word time='6.07'> del </word>
        <word time='6.54'> servicio </word>
        <word time='7.25'> ferroviario </word>
        <word time='7.70'> magdalena </word>
        <word time='8.19'> alvarez </word>
        <word time='8.88'> ministra </word>
        <word time='9.02'> de </word>
        <word time='9.54'> fomento </word>
</recognized>

REINITIATING RECOGNIZER
```

Figura 5.6: *Resultado del reconocedor*

La transcripción original es la siguiente:

“Se espera que en un plazo aproximado de 48 horas se conozcan todos los detalles del siniestro y que pueda restablecerse mañana el operativo del servicio ferroviario. Magdalena Álvarez es la ministra de fomento.”

La transcripción con la temporización es la salida final del sistema completo, lo que usaremos como BD. Al texto pueden añadirse más etiquetas de tener, como ya se ha comentado, un reconocedor del hablante.

El MG coge los documentos de esta Base de Datos y solo podrá encontrar las transcripciones que hayan sido subidas al archivo, tras el análisis del audio correspondiente. Para la búsqueda de archivos sonoros, hemos confeccionado una BD de prueba con transcripciones de las noticias que se emitieron en el informativo del mediodía el 22/08/2006.

5.3 Interfaz de usuario

Figura 5.7: *Buscador.html*

El interfaz de usuario es un interfaz web muy sencillo (ver fig. 5.7 y 5.8) consistente en un buscador, donde se introduce el texto clave que describe el audio a localizar, y la página de audios encontrados, resultante de la acción buscar.

El buscador se compone de dos categorías: búsqueda por texto y por locutor. En el primer recuadro indicamos qué se dice y en el segundo quién lo dice. De modo que si queremos buscar todos los audios de tal locutor, pondremos su nombre en el cajetín correspondiente de búsqueda (solo se muestran los 10 primeros, límite configurado en el MG). Si no indicamos locutor, se muestran todos los audios sobre el tema puesto en la descripción; y si indicamos ambos, tendremos en cuenta solo los resultados que cumplen ambas restricciones.

Para escuchar los audios basta con hacer clic en *play*. Pero no cualquier navegador es compatible con el código SMIL, utilizado para reproducir fragmentos de los clips de audio disponibles. Por eso la solución final adaptada ha sido utilizar simplemente javascript para el control del audio.

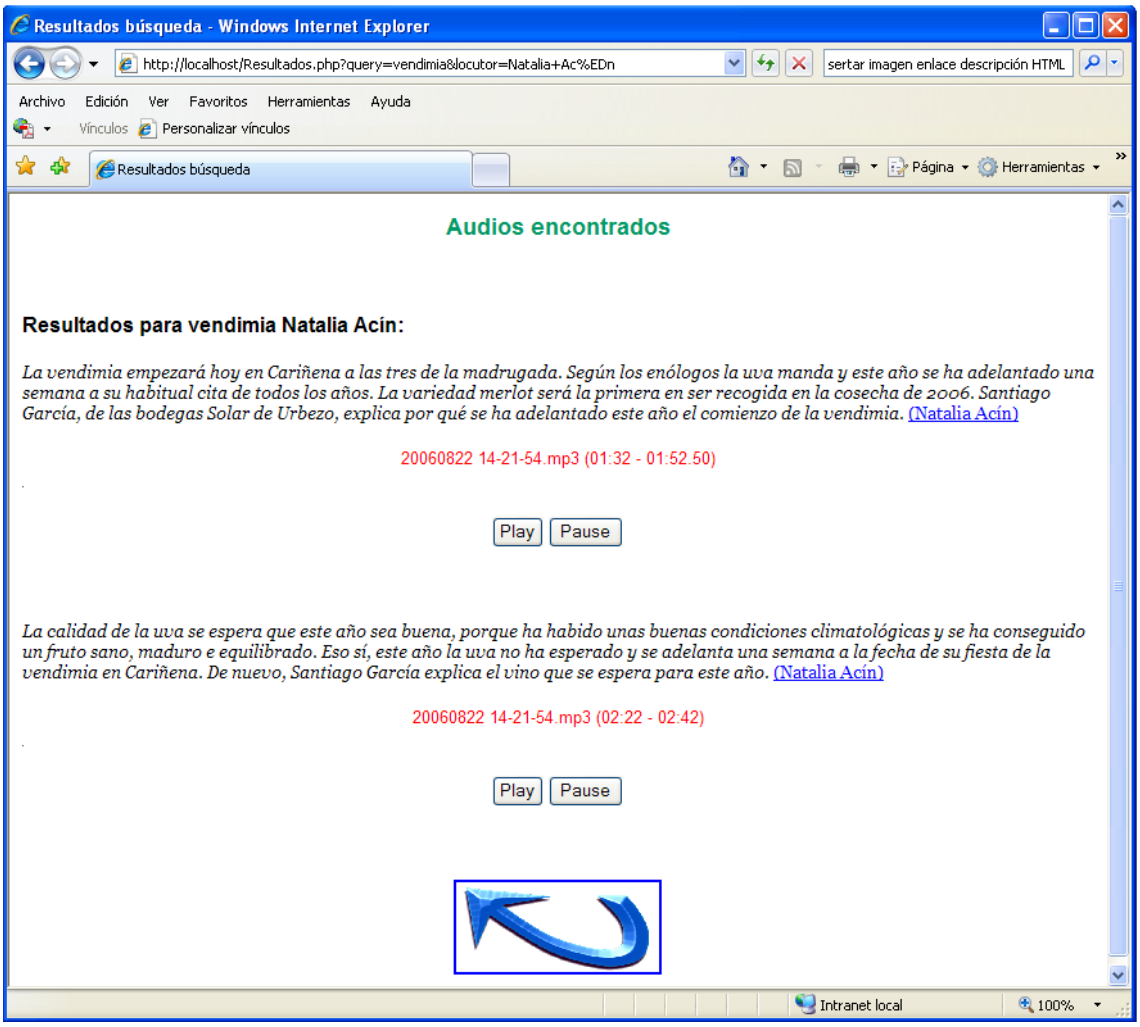

Figura 5.8: *Resultados.php*

Además, la solución inicial presentaba otro pequeño inconveniente que en modo local de pruebas no se apreciaba, y es que si el servidor está en otra máquina (algo habitual), la reproducción tardaba en iniciarse ya que primero se cargaba todo el fichero y después se reproducía el fragmento indicado. Para evitarlo, se ha confeccionado un script en Linux, donde se encuentra la base de datos, que fragmenta los audios originales en los segmentos resultantes del análisis.

La página de resultados muestra entre paréntesis el locutor que narra la noticia. Si hacemos clic encima del nombre, se abre una nueva ventana con la información que de ellos hay en Internet. Los locutores principales tienen su página de presentación dentro de la web de Aragón Radio. Los reporteros eventuales, que trabajaron temporalmente, carecen de esta información pero pulsando sobre el nombre accedemos directamente a su búsqueda en la red.

6 Conclusiones

Para finalizar, vamos a enumerar las conclusiones extraídas de la ejecución del proyecto. Repasaremos las dificultades encontradas y las soluciones dadas, las aplicaciones de la herramienta desarrollada y su importancia, así como las ampliaciones y mejoras que puede experimentar en un trabajo futuro.

6.1 Revisión final de todo el PFC

El objetivo a cumplir era montar la estructura de un sistema de búsqueda de documentos hablados e ilustrar con un ejemplo. La clave estaba en conseguir la ingesta del audio que permitiese la creación de una adecuada Base de Datos donde poder fácilmente encontrar los archivos buscados. La problemática inicial era la siguiente: en la actualidad se disponía de buscadores de texto bastante precisos pero se carecía de herramientas que extendieran esa búsqueda a todo tipo de archivos multimedia.

Para solventar este problema, se ha llevado a cabo un análisis del audio que permite reducir toda su información relevante a un documento de texto y así poder aplicar cualquiera de los motores de búsqueda desarrollados para la implementación del buscador web.

Al final, el resultado son pequeños fragmentos, con cada una de las intervenciones de un locutor, transcritos y con las etiquetas temporales de inicio de cada palabra. Pero los archivos a analizar no pueden ser los que de entrada se disponen.

El Archivo Sonoro con el que contamos se compone de audios de 15 min cada uno, cuyo contenido es la emisión de radio del día y hora que marca el nombre del archivo. Esta extensión es demasiado grande para poder ser analizada, ya que aparte de un costosísimo tiempo de procesado es inviable: los módulos del sistema no tienen capacidad para tratar archivos tan grandes. Es la gran limitación que presenta y viene dada por el reconocedor automático del habla. Este es el módulo más restrictivo porque implica el mayor coste computacional.

El problema radica en que el reconocedor utilizado es de pequeño/medio vocabulario, no preparado para la tarea. El motivo de su elección fue precisamente su simplicidad y su estabilidad. Mientras un reconocedor más complejo puede dar problemas (colgarse, fallo de fuera de memoria), este es sencillo y cómodo de utilizar y estaba ya listo para usarse aunque fuese limitado. El reconocedor más potente que se desarrolla en el área, ha estado en continuo cambio durante la realización de este proyecto. Además, no hay que olvidar que el objetivo principal era trabajar en la integración. Una vez montada la estructura, puede cambiarse fácilmente un reconocedor por otro.

Con el RAH actual, no se pueden procesar audios de un minuto o más. Tendremos que fragmentarlos cada 30s y transcribir frase a frase lo que dice cada locutor.

Queda pendiente lograr potentes reconocedores que consigan, con una precisión aceptable, transcripciones de audios de mayor duración, pues tener que subdividir los audios en fragmentos tan pequeños resta rentabilidad al proyecto, y el coste de hacer la trascripción automática o a mano sería prácticamente el mismo.

Este PFC es sólo una aproximación a la solución final que tendrá que tomarse, y que puede mejorarse añadiendo nuevos módulos de clasificación que tengan en cuenta los distintos entornos, hablantes e incluso estados de ánimo para que el RAH lleve a cabo un análisis más especializado y preciso.

6.2 Trabajo futuro

Este proyecto admite algunas mejoras que se pueden aplicar en un futuro:

1. **Inclusión del módulo speaker_id**, junto a otros módulos adicionales, que sirven para clasificar con más detalle el audio analizado. Cada segmento pertenece a un grupo (c_j) y cada una de estas agrupaciones c_j puede ser modelada con una distribución normal multivariable $N(\mu_j, \Sigma_j)$. Conocer cuando un hablante dado está produciendo voz dentro de un *stream*, puede ayudar a dirigir un modelo de adaptación para el reconocimiento de voz más efectivo para un hablante en particular.

2. **Tener en cuenta la segmentación en el módulo RAH** y obtener un formato de BD similar al utilizado de prueba, donde se indicaba el instante de inicio y fin del segmento transcrito dentro del clip de audio procesado. El resultado del RAH ahora se escribe en las pantallas de dos servicios: el propio RAH y la consola de usuario. La modificación a realizar sería, una vez implementada la estructura de la BD, que la consola escribiese en un documento todas las transcripciones con su marca temporal en vez de en pantalla, y esta transcripción más etiquetas sería el archivo a subir al clúster.

3. **Cambiar el reconocedor por uno de gran vocabulario adaptado para la tarea** que permitirá obtener con mayor precisión la transcripción del audio entrante. Esto, junto a pasarle el resultado de la segmentación, facilitará el análisis y eliminará el problema de tener que subdividir en fragmentos muy pequeños, pues transcribirá segmento a segmento. La tecnología actual solo puede realizar sistemas RAH que trabajen con un error aceptable en entornos semánticos restringidos.

El trabajo realizado se ha centrado en archivos de audio pero es extensible a cualquier archivo multimedia. Es otro de los frentes del trabajo futuro: hacer un buscador para vídeos, etc.

Bibliografía

[1] I. H. Witten, A. Moffat and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco, CA: Morgan Kaufmann, second edition, 1999.

[2] J. E. García y A. Ortega. "Arquitectura EDECAN. Jornada técnica sobre la arquitectura y los demostradores dentro del proyecto sd-team." Zaragoza, octubre de 2009.

[3] J. H. L. Hansen, R. Huang, B. Zhou, M. Seadle, J. R. Deller, Jr., A. R. Gurijala, M. Kurimo and P. Angkititrakul. "SpeechFind: Advances in Spoken Document Retrieval for a National Gallery of the Spoken Word", *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 712–730; September 2005.

[4] B. Zhou. *Audio parsing and rapid speaker adaptation in speech recognition for spoken document retrieval*. PhD Thesis, pp. 2–5; 2003.

[5] [Online] Disponible en: http://es.wikipedia.org/wiki/Socket_de_Internet

[6] S. Jothilakshmi, V. Ramalingam and S. Palanivel. "Unsupervised speaker segmentation with residual phase and MFCC features", *Expert Systems with Applications*, vol. 36, pp. 9800; 2009.

[7] S. Lucas. Proyecto Fin de Carrera (apdo. 2.1.2), Escuela Politécnica Superior, Universidad Autónoma de Madrid, septiembre de 2008.

[8] J. Ramírez, J. M. Górriz and J. C. Segura. "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness", *Robust Speech Recognition and Understanding*, I-Tech, Vienna, Austria, June 2007.

[9] J. Ramírez, J. C. Segura, C. Benítez, Á. de la Torre and A. Rubio. "Efficient Voice Activity Detection Algorithms Using Long-Term Speech Information", *Speech Communication*, vol. 42, no. 3-4, pp. 271–287; 2004.

[10] J. Ramírez, J. C. Segura, C. Benítez, Á. de la Torre and A. Rubio. "Voice Activity Detection with Noise Reduction and Long-Term Spectral Divergence Estimation", *ICASSP 2004*, vol. 2, no. 29, pp. 1093–1096; Montreal, May 2004.

[11] R. Huang and J. H. L. Hansen. "Advances in Unsupervised Audio Segmentation for the Broadcast News and NGSW Corpora", *ICASSP 2004*, vol. 1, no. 29, pp. 141–144; Montreal, May 2004.

[12] R. Huang and J. H. L. Hansen. "Advances in Unsupervised Audio Classification and Segmentation for the Broadcast News and NGSW Corpora", *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 3, pp. 907–919; May 2006.

[13] B. Zhou and J. H. L. Hansen. "Efficient Audio Stream Segmentation via the Combined T^2 Statistic and Bayesian Information Criterion", *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 4, pp. 467–474; July 2005.

[14] E. Lleida. "Reconocimiento Automático del Habla", octubre de 2000. [Online] Disponible en: http://dihana.cps.unizar.es/investigacion/voz/rah.html

[15] M. F. Porter. "Snowball: A language for stemming algorithms", October 2001. [Online] Available: http://snowball.tartarus.org/texts/introduction.html

[16] Snowball Project (1999). "A Spanish Stemming Algorithm". [Online] Available at http://snowball.tartarus.org/algorithms/spanish/stemmer.html

[17] [Online] Disponible en: http://es.wikipedia.org/wiki/Java_Servlet

Lista de figuras