INSTITUTO TECNOLÓGICO DE CD. MADERO

División de Estudios de Posgrado e Investigación

"POR MI PATRIA Y POR MI BIEN"

TESIS

Un Sistema para la Identificación Automática de Voces Usando Características Acústicas

PARA OBTENER EL TÍTULO DE:

Maestro en Ciencias en Ciencias de la Computación

Presenta:

I. S. C. Jesús Eduardo Carrillo Ibarra

Director de Tesis:

Dr. Arturo Hernández Ramírez

Co-Director

M. C. José Apolinar Ramírez Saldivar

Cd. Madero, Tamaulipas, mayo de 2013

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Subsecretaría de Educación Superior
Dirección General de Educación Superior Tecnológica
Instituto Tecnológico de Ciudad Madero

"2013, Año de la Lealtad Institucional y Centenario del Ejército Mexicano"

Cd. Madero, Tamps; a 13 de Marzo de 2013.

OFICIO No.: U5.103/13
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN
DE TESIS

C. ING. JESÚS EDUARDO CARRILLO IBARRA
PRESENTE

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestría en Ciencias en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"UN SISTEMA PARA LA IDENTIFICACIÓN AUTOMÁTICA DE VOCES USANDO CARACTERÍSTICAS ACÚSTICAS"

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

ATENTAMENTE
"Por mi patria y por mi bien"

M.P. MARÍA YOLANDA CHÁVEZ CINCO
JEFA DE LA DIVISIÓN

S.E.P.
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION
I T C M

c.c.p.- Archivo
Minuta

MYCHC 'NLCO 'jar

Ave. 1° de Mayo y Sor Juana I. de la Cruz, Col. Los Mangos, CP. 89440 Cd. Madero, Tam.
Tel. (833) 357 48 20, Fax, Ext. 1002, e-mail: itcm@itcm.edu.mx
www.itcm.edu.mx

ISO 9001 CERTIFIED COMPANY
ISO 14001 CERTIFIED COMPANY

Declaración de Originalidad

Declaro y prometo que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y publicaciones.

Además, en caso de infracción de los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad por la infracción y relevo de ésta a mi director de tesis y codirectores de tesis, así como al Instituto Tecnológico de Ciudad Madero y sus autoridades.

9 de Mayo del 2013, Cd. Madero, Tam.

Ing. Jesús Eduardo Carrillo Ibarra

Dedicatoria

Dedico este trabajo de investigación y desarrollo a mis papás, Leticia Ibarra García y Juan José Carrillo Martínez, así como a mi hermano Juan José Carrillo Ibarra y a mi nueva hermana y cuñada Flor Elizeth Nicolás Hernández y mi querido sobrino Juan José Carrillo Nicolás.

Agradecimientos

Agradezco primeramente a Dios por la oportunidad y el privilegio de permitirme seguir estudiando y preparándome profesionalmente.

Agradezco a mi familia por su apoyo incondicional en mis estudios.

Agradezco a mi director de tesis Dr. Arturo Hernández Ramírez y a mi Co-Director M. C. José Apolinar Ramírez Saldivar por sus grandes y pacientes explicaciones de temas que solamente con su ayuda he podido entender.

Agradezco a mi comité tutorial por sus valiosas y oportunas observaciones y revisiones de este trabajo.

Finalmente agradezco a todos mis compañeros Vicky, Julio, Raymundo, Alejandro, Daniel y Carlos, por su ayuda y apoyo durante las clases, así como por todos los momentos de alegría y risas que vivimos juntos.

Resumen

La identificación automática del lenguaje es la tarea de determinar el idioma que se habla a partir de una muestra corta de voz. Sus aplicaciones se enfocan principalmente a la identificación del lenguaje como preprocesamiento en sistemas muntilingüistas, tales como sistemas de soporte, emergencias o traducciones.

Los principales enfoques para resolver la tarea LID son: acústico, prosódico y fonotáctico, siendo este último el más popular y el que mejores resultados obtiene. El presente trabajo, al igual que los trabajos antecedentes, utiliza un enfoque puramente acústico. Así mismo, se utiliza la transformada wavelet como principal herramienta para caracterizar la señal del habla.

El objetivo principal de este trabajo es desarrollar un prototipo de software que realice la Identificación Automática del Lenguaje con un enfoque puramente acústico utilizando el trabajo de clasificación y multiclasificación en el entorno de MATLAB.

Summary

The automatic language identification is the task of identifying the spoken language in a short voice sample. Its applications are mainly focused on the identification of language as preprocessing in multilingual systems, such as support, emergency or translations systems.

The main approaches solving the task LID are: acoustic, prosodic and phonotactic, being phonotactic approach the most popular and best performing. This work, like the works history, used a purely acoustic approach, likewise, the wavelet transform is used as the main tool to characterize the speech signal.

The main objective of this work is to develop a software prototype that performs the automatic language identification with a purely acoustic approach using classification and multi classification in the MATLAB environment.

Índice de Contenido

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

INTRODUCCIÓN

La identificación automática del lenguaje (LID) es la tarea de identificar automáticamente una lengua a partir de una muestra corta de habla [Navrátil 2006]. Esta tarea toma especial importancia en la actualidad debido a la globalización y a la constante migración regional de hablantes de alguna lengua indígena en países como México. Además, este tema de investigación se ha estado desarrollando permanentemente como lo muestran los trabajos [Schultz-Kirchhoff 2006] [Mariani 2009] y [Timoshenko 2011].

La presente investigación, tiene como fin principal consolidar los trabajos precedentes [Vargas Martínez 2009], [Medina Trejo 2011] y [Hernández Zepeda 2011] en un prototipo de sistema de Identificación del Lenguaje que solamente utilice características acústicas. Los trabajos anteriores básicamente se dedicaron a realizar clasificación y multiclasificación de lenguas. En el presente trabajo, lo que pretendemos hacer es Identificación de Lenguas, lo cual significa que a partir de una muestra corta de habla se determine la lengua que se habla en dicha muestra.

1.1 Objetivos

1.1.1 Objetivo General

Desarrollar un sistema que realice la Identificación Automática del Lenguaje a partir de una muestra corta de audio.

1.1.2 Objetivos Específicos

- Implementar en el entorno de MATLAB la generación de modelos matemáticos de los idiomas soportados por el sistema.

- Desarrollar una interfaz visual que permita agregar más idiomas según sea necesario.
- Desarrollar un sistema web que realice la Identificación del Lenguaje.
- Determinar la mejor configuración del sistema para la realizar la identificación de idiomas.

1.2 Justificación

Las condiciones actuales de migración, tanto internacional como regional, hacen que la Identificación de una lengua se convierta en una actividad sumamente importante para brindar atención y servicios adecuadamente. Es por esto que se hace necesario culminar las investigaciones antecedentes con una integración final en forma de una aplicación para computadora que permita Identificar el Lenguaje en forma automática.

1.3 Definición del problema.

Los trabajos anteriores permitieron contar con una metodología estable y funcional para clasificar idiomas. Sin embargo, la LID contempla una identificación real del idioma en muestras de habla. Esto hace necesario adaptar la metodología anterior para que realice una identificación real del idioma en muestras cortas de habla y así, poder generar un primer prototipo de software LID acústico.

1.4 Antecedentes del Proyecto

El presente proyecto tiene como antecedentes la tesis de doctorado [Reyes – Herrera 2007], realizada en el Instituto Nacional de Astrofísica, Óptica y Electrónica, las tesis de maestría [Vargas – Martínez 2008 y Medina - Trejo 2011] y la tesis de licenciatura [Hernández - Zepeda 2011] desarrolladas en el Instituto Tecnológico de Ciudad Madero.

A continuación se presenta un breve resumen de dicho trabajos.

1.4.1 Reyes-Herrera

En el trabajo de Reyes Herrera se siguen básicamente las investigaciones de Cummins [Cummins et al., 1999] y Rouas [Rouas et al. 2003], ambas basadas en el modelado prosódico. Su principal aportación es el uso de la transformada wavelet para caracterizar las señales del habla.

En su experimentación, se contemplan pruebas de clasificación usando las muestras de audio de la base de datos OGI TS [Yeshwant Muthusamy et al. 1992]. En dichas pruebas, se toman 10s y 45s de cada muestra para formar los modelos de los 9 idiomas, obteniendo 94 y 86 por ciento de clasificación por validación cruzada respectivamente.

1.4.2 Vargas Martínez

Vargas Martínez siguió la misma línea de investigación de Reyes Herrera, el uso de la transformada Wavelet para caracterizar la señal de habla. El aporte de Vargas Martínez a la investigación de Reyes Herrera, es que él contempla cuestiones no vistas anteriormente, por ejemplo, el tiempo necesario para una aplicación real. Una aplicación real necesita una cantidad corta de habla para identificar el lenguaje en un tiempo razonable (Reyes Herrera necesita 50 segundos, más el procesamiento). También se utilizó un módulo de detección de voz activa, para la eliminación de pausas largas. Además, agrega medidas estadísticas sobre los coeficientes wavelet, a las cuales aplica ganancia de información con el fin de reducir la dimensionalidad de la información.

1.4.3 Medina Trejo y Hernández Zepeda

Los principales aportes de Medina Trejo y Hernández Zepeda son el diseño de una metodología de multiclasificación y la aplicación de 2 nuevas medidas estadísticas para la caracterización de la entrada de audio.

La metodología diseñada por Medina Trejo y Hernández Zepeda se basa principalmente en la metodología usada por Vargas Martínez ya que se utilizan los biclasificadores resultantes de ésta para realizar la multiclasificación.

La multiclasificación desarrollada en este trabajo realiza una clasificación por torneos entre los biclasificadores. Esto es, tomando los biclasificadores se realiza una competencia entre ellos para determinar una clase ganadora. Las competencias se hicieron siguiendo las reglas de dos torneos principalmente, Wimbledon y Round Robin [Xia 2005].

Las medidas estadísticas agregadas para caracterizar la entrada de audio fueron la *kurtosis*[1] y la *skewness*[2]. El uso de estas dos medidas estadísticas mostró una mejora en los resultados.

Otro aporte, es la experimentación con distintos clasificadores. De la experimentación realizada, el clasificador Naive Bayes fue el que obtuvo mejores resultados. Además, se aportó experimentación con la familia de wavelets Dbn, con n>2, en la que los resultados obtenidos no son mejores que los obtenidos por Vargas Martínez, lo cual indica que la familia Dbn tiene un comportamiento similar.

1.5 Alcances y Limitaciones

A continuación se listan los alcances y limitaciones de este proyecto

- Se usarán las bases de datos OGI TS [Yeshwant Muthusamy et al. 1992] y Lwazi [Lwazi 2011].
- Se usarán solamente características acústicas.
- Se utilizará como base las metodologías desarrolladas en los trabajos antecedentes.
- Se contempla el desarrollo de un prototipo del sistema final.

[1] La kurtosis es un parámetro estadístico para medir la naturaleza gaussiana de los datos.
[2] La skewness es una medida de simetría, o más precisamente, de falta de simetría.

1.6 Organización del Proyecto

A continuación se describe brevemente el contenido en cada uno de los siguientes capítulos.

- Capítulo 2 Marco Teórico: En este capítulo se presentan los fundamentos teóricos sobres los cuales se fundamenta el proyecto.
- Capítulo 3 Estado del Arte: En esta sección se presenta una visión general de los trabajos relacionados con la Identificación Automática del Lenguaje.
- Capítulo 4 Metodología: Esta parte del documento describe la metodología utilizada para la elaboración de los modelos matemáticos y la identificación del lenguaje.
- Capítulo 5 Experimentación y Resultados: Se muestran el conjunto de pruebas y experimentación realizadas tanto para validar la creación de los modelos matemáticos como las pruebas para evaluar la identificación de nuevas instancias.
- Capítulo 6 Conclusión y Trabajos Futuros: Se presentan las conclusiones obtenidas durante el desarrollo del proyecto, así como las propuestas de trabajos para realizar en el futuro.
- Referencias.
- Anexos: Se muestran los códigos utilizados para implementar y adaptar las metodologías de este proyecto.

CAPÍTULO 2

MARCO TEÓRICO

2.1 Wavelets

Durante el desarrollo de este proyecto, se tomaron como base los libros "Foundations of Signal Processing" [Vetterli 2011], "Signal Processing for Communications" [Vetterli 2008] y "Wavelets and Subband Coding" [Vetterli 1995], para desarrollar los temas relacionados con los wavelets.

2.1.1 Historia de los Wavelets

La primer wavelet fue modelada por [Haar 1910] a principios del siglo pasado. Pero la construcción de wavelets de una manera más general para formar bases de funciones cuadrado integrables fue investigada en la década los años 80, con algoritmos eficientes para calcular la expansión de la señal. Al mismo tiempo, florecieron aplicaciones de estas técnicas en el procesamiento de señales.

Mientras que las expansiones lineales de funciones son temas clásicos, las formulaciones recientes contienen nuevas e interesantes características. Por ejemplo, los wavelets proporcionan buena resolución en tiempo y frecuencia, por lo tanto permiten ver "el bosque y los árboles". Esta característica es importante para el análisis de señales no estacionarias. Mientras que las funciones base de Fourier están dadas en una forma cerrada (senos y cosenos), muchas wavelets pueden ser obtenidas solamente a través de procedimientos computacionales. Aunque esto podría parecer un inconveniente, resulta que si uno está interesado en implementar una expansión de una señal en información real, entonces un procedimiento computacional es mejor que una expresión cerrada.

El interés en este tipo de expansiones se debe a la convergencia de ideas de distintos campos de investigación, y al hecho de que técnicas desarrolladas independientemente en estos campos pueden ser incluidas en un marco de referencia común.

El significado actual del nombre wavelet es debido a J Goupillaud, J. Morlet y A. Grossmann [Goupillaud - Morlet 1984] [Morlet – Grossmann 1989]. En el contexto de procesamiento de señales geofísicas ellos investigaron una alternativa al análisis local de Fourier basada en una función prototipo, y sus escalas y corrimientos. La modulación por exponenciales complejas en la transformada de Fourier es remplazada por una operación de escalamiento, y la noción de escala remplaza a la noción de frecuencia. La simplicidad y elegancia del esquema wavelet era atractivo y los matemáticos comenzaron a estudiar el análisis wavelet como una alternativa al análisis de Fourier. Esto llevó al descubrimiento de wavelets que forman bases ortonormales para espacios de funciones cuadrado integrables y otras por [Meyer 1990], [Daubechies 1998], [Battle 1987], [Lemarie 1988]. La formalización de estos trabajos realizada por [Mallat 1989] y [Meyer 1990] produjo un marco de referencia para las expansiones wavelet, llamado análisis multiresolución.

Por supuesto, estos logros fueron precedidos por una evolución de largo plazo desde la wavelet de Haar hasta el uso de la octava división del espectro de Fourier.

Paralelamente a los avances en matemáticas puras y aplicadas, se tuvieron avances en el procesamiento de señales, en el contexto de señales discretas en el tiempo. Impulsado por las aplicaciones como compresión de imágenes y del habla, el método llamado codificación subbanda (Subband Coding) fue propuesto por [Croisier 1976] usando una clase especial de filtro llamado filtro de cuadratura espejo (QMF) a finales de los 70’s. Esto llevó al estudio de bancos de filtro de reconstrucción perfecta, un problema resuelto en los 80’s por algunas personas, incluyendo [Smith 1984], [Barnwell 1986], [Mintzer 1985] y [Papoulis 1962].

En visión por computadora, las técnicas de multiresolución han sido usadas para varios problemas, desde estimación del movimiento hasta el reconocimiento de objetos [Rosenfeld 1984]. Las imágenes son aproximadas sucesivamente empezando desde una versión burda

hasta una versión fina. En particular, [Burt 1987] propuso dicho esquema para codificación de imágenes a principios de los 80's, llamándolo codificación piramidal. Este método es similar a la codificación subbanda. La vista de aproximaciones sucesivas es similar al análisis multiresolución usado en esquemas wavelets.

En gráficos de computadora, un método llamado refinamiento sucesivo interpola iterativamente curvas o superficies, y el estudio de dichas interpolaciones está relacionado con la construcción de wavelets y bancos de filtros [Cavaretta 1991], [Dyn 1990].

Finalmente, muchos procedimientos computacionales usan el concepto de aproximación sucesiva, algunas veces alternando entre resoluciones finas y burdas. Por ejemplo, los métodos multimalla utilizados para la solución de ecuaciones diferenciales parciales [Briggs 1987].

De hecho, quizás una de las grandes contribuciones de los wavelets ha sido juntar personas de diferentes campos, y a partir de esta unión y el intercambio de ideas, se ha progresado en varias áreas.

2.1.2 Transformada Wavelet

2.1.2.1 Transformada Wavelet Continua

La transformada wavelet es una técnica relativamente nueva que es considerada por los investigadores como una poderosa herramienta en el análisis de señales. Al igual que la Transformada de Fourier de Tiempo Corto (STFT), esta transformada utiliza una función ventana que encuadra una señal dentro de un intervalo de tiempo y focaliza el análisis sobre ese segmento de la señal.

La transformada wavelet continua intenta expresar la señal $x(t)$ mediante una expansión de términos, o coeficientes, proporcionales al producto interno de la señal y diferentes versiones escaladas y trasladadas de una función prototipo $\psi(t)$, más conocida como wavelet madre.

Asumiendo que tanto la señal, como la nueva $\psi(t)$ son de energía finita, podemos definir

$$CWT(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi\left(\frac{t-b}{a}\right) dt \tag{1}$$

como la Transformada Wavelet Continua. Ésta transformada también puede ser escrita en términos de la transformada de Fourier donde $X(\omega)$ y Ψ corresponde a la transformada de Fourier de $x(t)$ y ψ respectivamente:

$$CWT(a,b) = \frac{1}{2\pi\sqrt{a}} \int_{-\infty}^{\infty} X(\omega)\Psi(a\omega)e^{-i\omega b} dt \tag{2}$$

Como se puede observar han aparecido dos nuevas variables *a* y *b*. La variable *a* controla el ancho o soporte efectivo de la función ψ, y la variable *b* nos da la ubicación en el dominio del tiempo de ψ.

Ahora bien, para que este análisis sea posible y además para poder lograr una perfecta reconstrucción de la señal a partir de la transformada, la función $\psi(t)$ debe cumplir con la condición de admisibilidad [Chui 97] de la cual se desprende que

$$\Psi\,(0) = 0 \tag{3}$$

donde $\Psi = \Psi(\omega)$ corresponde a la transformada de Fourier de $\psi(t)$. El cumplimiento de esta condición significa que el valor medio de ψ es igual a 0, lo que a su vez implica

obligatoriamente que ψ tenga valores tanto positivos como negativos, es decir, que sea una onda. Además, como ψ es una función que "ventaniza" la señal sobre un intervalo de tiempo dado por *a* alrededor de un punto *t = b* se observa intuitivamente que es de soporte compacto, es decir, es una onda definida sobre un intervalo de tiempo finito, y esto es el porqué de su nombre *wavelet* u ondita.

Mediante la variable de escala nosotros podemos comprimir (|a| < 1) o dilatar (|a| > 1) la función $\psi(t)$, lo que nos dará el grado de resolución con el cual estemos analizando la señal. Por definición la Transformada Continua Wavelet es más una representación tiempo - escala que una representación tiempo - frecuencia. En particular, para valores pequeños de *a* la CWT obtiene información de $x(t)$ que está esencialmente localizada en el dominio del tiempo mientras que para valores grandes de *a* la CWT obtiene información de $X(\omega)$que está localizada en el dominio de la frecuencia. En otras palabras, para escalas pequeñas la CWT nos entrega una buena resolución en el dominio del tiempo mientras que para escalas grandes la CWT nos entrega una buena resolución en el dominio de la frecuencia. Cuando *a* cambia, tanto la duración como el ancho de banda de la wavelet cambian pero su forma se mantiene igual. En lo anteriormente dicho se encuentra la diferencia principal entre la CWT y la STFT, ya que la primera ocupa ventanas de corta duración para altas frecuencias y ventanas de larga duración para bajas frecuencias mientras que la STFT ocupa una sola ventana con la misma duración tanto para altas frecuencias como para bajas frecuencias.
Aunque la CWT trabaja con el término escala en vez de frecuencia, es posible mediante una constante $c > 0$ realizar un cambio de variable de una escala *a* a una frecuencia *ω* de la forma

$$a \to \omega = \frac{c}{a}$$

Donde *c* recibe el nombre de constante de calibración en unidades de frecuencia (tal como Hz). Con este cambio de variable podemos observar que la CWT localiza tanto la señal $x(t)$ en el dominio del tiempo como su espectro $X(\omega)$ en el dominio de la frecuencia en forma simultánea.

La variable b controla la ubicación de la función en el espacio de tiempo permitiéndonos deslizar $\psi(t)$ sobre el intervalo de tiempo en el que se haya definido $x(t)$. Un punto importante es que la función wavelet se traslada cubriendo toda la señal para cada valor de a, es decir, si la escala escogida es pequeña habrá más traslaciones de ψ que si la escala escogida es grande. Por lo tanto, la variable b nos da la cantidad por la $\psi\left(\frac{t}{a}\right)$ ha sido trasladada en el dominio del tiempo.

La continuidad de la CWT reside en que tanto la variable de escala como la variable de traslación varían en forma continua. Sin embargo, en términos de cálculo computacional es imprescindible discretizar la transformada, y la suposición más lógica es que tanto los valores de escala como traslación sean discretos.

2.1.2.2 Transformada Wavelet Discreta

Los sistemas wavelet discretos son todos aquellos que sean generados solo por traslaciones enteras y escalamientos de una única función wavelet $\psi(t)$, llamada wavelet madre o wavelet generadora, que da a luz a una familia de funciones de la forma:

$$\psi_{j,k}(t) = 2^{\frac{j}{2}}\psi\left(2^j t - k\right) \quad j,k \in \mathbb{Z} \tag{4}$$

Donde el factor $2^{\frac{j}{2}}$ mantiene una norma constante independiente de la escala j. Esta familia de funciones es llamada conjunto de expansión wavelet. La wavelet madre $\psi(t)$, trae siempre asociada consigo una función de escala $\phi(t)$. Con estas dos funciones podremos aproximar cualquier función o señal $f(t) \in L^2(\mathbb{R})$, mediante una de las funciones o mediante ambas, de la forma

$$\sum_k \sum_j c_{j,k}\phi(t) + \sum_k \sum_j d_{j,k}\psi(t) \tag{5}$$

El conjunto de expansión wavelet no es único, existen muchos y muy diferentes sistemas wavelet, pero todos tienen las siguientes características [Tamara 00]:

1. Un sistema wavelet puede describirse de una manera "amigable", como un conjunto de ladrillos (que para cada sistema pueden tener diferente forma) que sirven para reconstruir o representar una señal o función. Este conjunto es una expansión bi-dimensional, que suele ser una base para alguna clase de señal de una o más dimensiones. En otras palabras, si el conjunto de expansión está dado por $\{\psi_{j,k}(t)|j,k\ \in Z\}$, una expansión lineal puede ser

$$f(t) = \sum_k \sum_j d_{j,k}\psi_{j,k}(t)$$

2. La expansión wavelet entrega una localización tiempo-frecuencia instantánea de la señal, representación que puede explicarse como un pentagrama musical, donde la localización y forma de la figura musical nos dice cuando ocurre el tono y cuál es su frecuencia. Esto quiere decir que la mayor parte de la energía de la señal es bien representada por unos pocos coeficientes $d_{j,k}$. Mientras un coeficiente de Fourier representa un componente que dura todo el tiempo en que se extiende la señal, un coeficiente de expansión wavelet representa un componente bien definido en un intervalo de tiempo, esto es, un coeficiente wavelet es en sí bien localizado en el tiempo.
3. Los sistemas wavelet satisfacen las condiciones de multiresolución. Esto significa que si un conjunto de señales puede ser representado por una suma de $\phi(t-k)\ k\ \in Z$, un conjunto más amplio de señales (que incluye el conjunto original) puede ser representado por una suma $\phi(2t-k)\ k\ \in Z$.
4. Los coeficientes de más baja resolución pueden ser calculados a partir de los coeficientes de más alta resolución, mediante un algoritmo en forma de árbol, llamado banco de filtros. Esto permite un muy eficiente cálculo de los coeficientes de expansión (también conocida como la Transformada Wavelet Discreta).

5. El tamaño de los coeficientes de expansión wavelet disminuye rápidamente con *j* y *k*.
6. Los wavelets son ajustables y adaptables. Debido a que existen muchos wavelet, estos pueden ser diseñados para adaptarse a una aplicación particular.
7. La generación de wavelets y el cálculo de la Transformada Discreta Wavelet es bien realizada por una computadora, pues estos cálculos se remiten sólo a multiplicaciones y sumas.

2.1.2.3 Wavelet Daubechies Db2

¿Cómo se encuentra una wavelet? La idea principal es la auto similitud. Se empieza con la función $\phi(x)$ que se hace a partir de pequeñas versiones de sí misma. Los coeficientes a_k de la ecuación de refinamiento son llamados coeficientes filtro o mascaras. La función $\phi(x)$ es llamada función escala (o wavelet padre). Bajo ciertas circunstancias produce un wavelet.

$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \, \phi(2x - k)$$

Primero, se escoge la función escala para preservar el área en cada iteración, esto es $\int_{-\infty}^{\infty} \phi(x)dx = 1$. Integrando la ecuación de refinamiento tenemos

$$\int_{-\infty}^{\infty} \phi(x)dx = \sum a_k \int_{-\infty}^{\infty} \phi(2x - k)dx = \frac{1}{2}\sum a_k \int_{-\infty}^{\infty} \phi(u)du$$

Por lo tanto $\sum a_k = 2$. De modo que la estabilidad de la iteración forza la condición en los coeficientes de a_k . Segundo, la convergencia de la expansión wavelet requiere $\sum_{k=0}^{N-1}(-1)^k \, k^m a_k = 0$ donde *m* = 0, 1, 2,…, $\frac{N}{2} - 1$. Tercero, la condición de ortogonalidad de los wavelets requiere que $\sum_{k=0}^{N-1} a_k \, a_{k+2m} = 0$ donde *m* = 0, 1, 2,…, $\frac{N}{2} - 1$. Finalmente,

si se requiere que la función escala sea ortogonal $\sum_{k=0}^{N-1} a_k^2 = 2$. En resumen, las condiciones son

$$\sum_{k=0}^{N-1} a_k = 2 \quad \text{Estabilidad.}$$

$$\sum_{k=0}^{N-1} (-1)^k k^m a_k = 0 \qquad \text{Convergencia.}$$

$$\sum_{k=0}^{N-1} a_k a_{k+2m} = 0 \quad \text{Ortogonalidad.}$$

$$\sum_{k=0}^{N-1} a_k^2 = 2 \quad \text{Ortogonalidad de las funciones escala.}$$

Esta clase de funciones wavelet está restringida, por definición, a ser cero fuera de un intervalo pequeño. Esto hace la propiedad de soporte compacto. La mayoría de las funciones wavelets, al ser graficadas, lucen extremadamente irregulares. Esto es debido al hecho de que la ecuación de refinamiento asegura que la función wavelet $\psi(x)$ no es diferenciable en todas partes. Las funciones que son usadas normalmente para realizar las transformada consisten de algunos conjuntos de coeficientes resultando en una función con una forma discernible.

Ahora veamos cómo generar el wavelet Daubechies. Considerando $N = 4$ para las restricciones mencionadas tenemos las siguientes ecuaciones

$$a_0 + a_1 + a_2 + a_3 = 2$$
$$a_0 - a_1 + a_2 - a_3 = 0$$
$$-a_1 + 2a_2 - 3a_3 = 0$$
$$a_0 a_2 + a_1 a_3 = 0$$

$$a_0^2 + a_1^2 + a_2^2 + a_3^2 = 2$$

Las soluciones para este sistema de ecuaciones son

$$a_0 = \frac{1+\sqrt{3}}{4}, a_1 = \frac{3+\sqrt{3}}{4}, a_2 = \frac{3-\sqrt{3}}{4}, a_0 = \frac{1-\sqrt{3}}{4}.$$

La wavelet correspondiente es Daubechies-2 (db_2), la cual tiene soporte compacto para el intervalo [0,3], como se muestra en la siguiente figura.

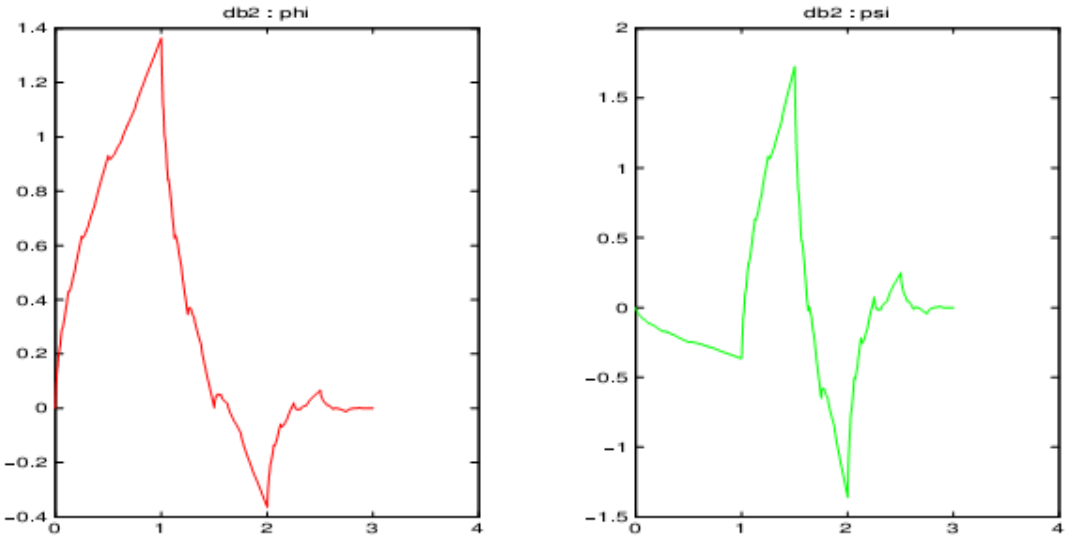

Figura 1 Wavelet Daubechies . 2

La construcción mostrada es conocida como la construcción wavelet Daubechies. En general, db_n representa la familia de wavelets Daubechies y *n* es el orden.

2.1.3 Concepto de Multiresolución

Una expansión ligeramente diferente se obtiene con pirámides multiresolución ya que la expansión es en realidad redundante (el número de muestras en la expansión es más grande que la señal original). Sin embargo, conceptualmente, está íntimamente relacionada con la codificación subbanda y la descomposición wavelet. La idea básica es una aproximación

sucesiva. Una señal se describe como una aproximación burda más una predicción de error basada en la versión burda. La reconstrucción es inmediata: simplemente se agrega la predicción al error de predicción. El esquema puede ser iterado sobre la versión burda. Esto puede ser visto como si el filtro de paso bajo cumpliera ciertas restricciones de ortogonalidad, entonces este esquema es idéntico a una serie wavelet discreta sobre muestreada. De lo contrario, el enfoque de aproximación sucesiva puede ser visto conceptualmente como idéntico a la descomposición wavelet ya que este realiza un análisis multiresolución de la señal.

La Figura 2 muestra un diagrama de la descomposición piramidal, con las imágenes resultantes. Después de la descomposición, se obtiene una imagen de resolución burda de la mitad de tamaño, así como una imagen de error de tamaño completo (por esto la redundancia). Para aplicaciones, la descomposición en una resolución burda que dé una aproximada pero adecuada versión de la imagen completa, más una diferencia o detalles de la imagen, es conceptualmente importante.

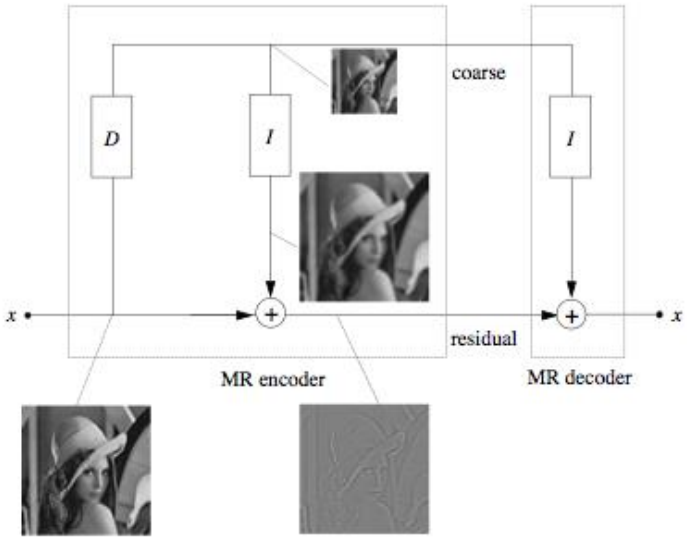

Figura 2 Descomposición piramidal de una imagen donde la codificación es mostrada a la izquierda y la descodificación es mostrada a la derecha. Los operadores D e I corresponden a los operadores de decimación e interpolación. Por ejemplo, *D* produce una imagen de *N*/2x*N*/2 a partir de una imagen de *N*x*N* original, mientras que I interpola una imagen produce una imagen *N*x*N* basado en la imagen original *N*/2x*N*/2.

Consideremos el siguiente problema práctico: Los usuarios quieren acceder y recuperar imágenes electrónicas de una base de datos usando una red de computadoras con ancho de banda limitado. Ya que los usuarios tienen una idea aproximada de cual imagen quieren, ellos buscan entre las imágenes antes de establecer la imagen final. Dado el ancho de banda limitado, es mejor navegar entre las imágenes con una versión burda de estas, la cual puede ser transmitida más rápido. Una vez que la imagen es elegida, el resto de la imagen puede ser enviado. El ejemplo anterior es una de los muchos escenarios en los que la descomposición multiresolución puede usarse en telecomunicaciones.

Las descomposiciones multiresolución son también importantes en las tareas de visión por computadora como segmentación de imágenes y reconocimiento de objetos: la tarea es realizada en forma de sucesiones aproximadas, empezando con una versión burda y después usando esto como una suposición para el resto de la tarea. Sin embargo, esta forma es un acercamiento voraz que suele ser subóptimo.

2.1.4 Algoritmos de descomposición y reconstrucción

En esta sección, se observan las operaciones estándar requeridas para pasar de la expresión de una función en bases canónicas V_J a su expresión en bases Haar e inversamente.

A. Sea f una función en $L^2(\Omega)$ y J un entero fijo arbitrario. Calculamos los 2^J coeficientes c_J^k, exactamente si tenemos una expresión para f, o aproximadamente usando una muestra de 2^J valores de f en intervalos de Ω_k^J . Empezando desde estos 2^J coeficientes c_J^k, calculamos sucesivamente los coeficientes c_j^k y d_j^k de acuerdo al siguiente algoritmo

for $j = J-1, \ldots, 1, 0$ **compute**
  for $k = 0, 1, \ldots, 2^j - 1$ **compute**
    $c_j^k = (c_{j+1}^{2k} + c_{j+1}^{2k+1})/\sqrt{2}$
    $d_j^k = (c_{j+1}^{2k} - c_{j+1}^{2k+1})/\sqrt{2}$
  end
end

Figura 3 Algoritmo de descomposición

Este cálculo es referido como el algoritmo de análisis o descomposición de *f*. Lo podemos representar el paso *j* de este algoritmo por esquema simbólico

$$\begin{array}{c} c_j^k \\ (0 \le k < 2^j) \\ \swarrow \quad \searrow \\ c_{j-1}^k \qquad\qquad d_{j-1}^k \\ (0 \le k < 2^{j-1}) \ (0 \le k < 2^{j-1}) \end{array}$$

Figura 4 Representación simbólica del paso *j* del algoritmo de descomposición.

Una vez calculados lo 2^j coeficientes d_j^k, estos ya no son usados en el siguiente paso del algoritmo de descomposición, solo los 2^j valores de los coeficientes $\{c_j^k\}_k$ son requeridos para calcular los coeficientes $\{c_{j-1}^k\}_k$ y $\{d_j^k\}_k$. El costo computacional del paso *j* en el algoritmo es el costo de calcular 2 x 2^{j-1} coeficientes, esto es, exactamente 2^{j+1} operaciones. El costo computacional del algoritmo de descomposición requerido para obtener los valores de 2^J coeficientes, es entonces

$$2^{J+1} + \cdots + 2^j + \cdots + 2^2 + 1 = 2^{J+2} = 4 \times 2^J.$$

Esto puede ser considerado como un valor óptimo, ya que estamos calculando 2^J salidas a partir de 2^J entradas para un costo de $O(2^J)$ operaciones

B. Inversamente, asumimos que conocemos c_0^0, el principal valor de *f* en Ω, y todos los coeficientes d_j^k, para *j*=0, . . ., *J*-1. Entonces obtenemos todos los coeficientes c_j^k usando el

siguiente algoritmo:

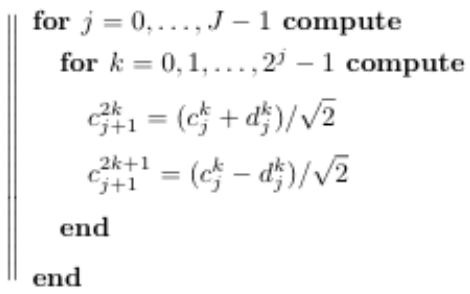

Figura 5 Algoritmo de reconstrucción.

Se resalta de nuevo, que el costo computacional del paso *j* de este algoritmo es igual al cálculo de 2 x 2^{j-1} coeficientes, esto es, 2^{j+1} operaciones.

2.2 Tópicos de Minería de Datos

Esta investigación hace uso de tópicos de Minería de datos, los cuales, en su mayoría, han sido desarrollados y explicados en los trabajos antecedentes [Vargas Martínez 2009], [Medina Trejo 2011] y [Hernández Zepeda 2011]. A continuación se explican solo aquellos conceptos utilizados en este trabajo y que no han sido explicados con anterioridad.

2.2.1 Validación cruzada

Estimar la precisión de un modelo creado a partir de un algoritmo de aprendizaje supervisado es importante no solo para predecir su precisión de futuras predicciones sino también para escoger un clasificador de un grupo dado.

La precisión de un clasificador es la probabilidad de clasificar correctamente instancias seleccionadas aleatoriamente. La validación cruzada es un método que sirve para calcular dicha precisión.

En la validación cruzada con *k* paquetes, el conjunto D es dividido aleatoriamente en *k* subconjuntos D_1, D_2,…, D_k, mutuamente exclusivos de aproximadamente el mismo tamaño.

El modelo es entrenado con el subconjunto $D \backslash D_t$ y evaluado con el subconjunto D_t k veces; cada vez $t \in \{1, 2, \ldots k\}$ [Kohavi 1995].

La proposición anterior nos ayuda a entender una posible suposición que se hace cuando se usa validación cruzada: si el modelo es inestable para un conjunto de datos bajo un conjunto de perturbaciones introducidas por la validación cruzada, la precisión de predicción será poco estable. Si el clasificador es estable para cierto conjunto de datos, podemos esperar una estimación estable.

2.3 Ritmo

El habla es percibida como una secuencia de eventos, y la expresión ritmo es usada para referirse a la manera en que estos eventos son distribuidos en el tiempo. El ritmo puede ser definido como una organización sistemática de unidades prominentes y menos prominentes del habla en el tiempo. La prominencia de estas unidades es expresada por frecuencias fundamentales más altas, mayor duración y/o intensidad más alta.

La percepción del ritmo en el habla ha sido objeto de interés entre los lingüistas por décadas. Este interés viene a partir de la observación de que los lenguajes tienen distintos tipos de ritmos.

[Pike 1945] sugirió que existen 2 tipos de ritmos: Uno es debido a la recurrencias de *stresses*, mientras que el otro es debido a la recurrencia de sílabas, dando la terminología "stress-timed" y "syllable-timed". Los lenguajes con ritmo stress-timed muestran patrones de igual duración entre sílabas prominentes, mientras que los lenguajes syllable-timed tienen sílabas de igual duración. [Abercrombie 1967] generalizó esta idea y además sugirió que todos los lenguajes pueden ser clasificados en una de estas clases. La teoría también dice que las dos categorías de ritmos son mutuamente exclusivas y que cada lenguaje es caracterizado ya sea por una o por otra de estos dos tipos de ritmos.

De acuerdo con [Abercrombie 1967], el ritmo de un lenguaje está relacionado con la fisiología de la producción del habla y puede ser definido como una combinación de pulsos del pecho y pulsos de tensión. Basado en estas notaciones, las clases de ritmo pueden ser caracterizadas como sigue:

Lenguajes stress-timed

- pulsos de tensión son igualmente espaciados
- no se puede medir isocronía entre intervalos de tensión

Lenguajes syllable-timed

- los pulsos del pecho son igualmente espaciados
- no se puede medir la isocronía entre la duración de las sílabas

[Ladefoged 1975] propuso una tercera clase de ritmo, llamada mora-timed, para lenguajes como el Japonés y el Tamil, donde el ritmo es determinado por unidades más cortas que las silabas, conocidas como morae. Tradicionalmente, morae son subunidades de sílabas que consisten de una vocal corta y precedida de una consonante. En lenguajes de tipo mora-timed, las morae sucesivas son dichas del tal forma que tengan casi la misma duración, lo que hace que estos lenguajes sean más similares a los lenguajes syllable-timed que a los lenguajes del tipo stress-timed.

CAPÍTULO 3

ESTADO DEL ARTE

En la actualidad, la Identificación Automática del Lenguaje (LID) es en un campo de investigación cada vez más estudiado, como lo muestran las publicaciones en esta área del conocimiento. En este capítulo, se describen los acercamientos comúnmente aplicados en la tarea LID: acústico, fonotáctico y prosódico. Asimismo, usaremos las siguientes referencias para elaborarlo: "Multilingual Speech Processing" [Navratil 2006], "Language and Speech Processing" [Mariani 2009] y la tesis doctoral de "Rhythm Information for Automated Spoken Language Identification" [Timoshenko 2011].

3.1 Sistemas LID

A continuación se presentan algunos sistemas de identificación automática del lenguaje. La descripción de algunos trabajos es breve ya que, al ser software propietario, la información acerca de su funcionamiento interno es restringida.

3.1.1 Nexidia

El software desarrollado por Nexidia [Nexidia 2012] comprende un conjunto de herramientas para automatizar el proceso de analizar un clip de audio para determinar con alto grado de exactitud qué lenguaje está siendo hablado en el clip. Estas herramientas proveen de una identificación rápida (velocidades en tiempo real), con la flexibilidad de permitir al usuario modificar el conjunto de modelos, para un ambiente de audio en particular y el número de lenguajes.

Además, las herramientas de Nexidia pueden identificar múltiples lenguajes dentro del mismo clip de audio.

El sistema desarrollado cuenta con un conjunto de librerías en C++ y COM que proveen acceso al núcleo de las API's de la identificación del lenguaje desde otros sistemas desarrollados en tecnologías compatibles. También cuenta con una herramienta de entrenamiento para crear los modelos para la identificación del lenguaje de 2 o más lenguajes de interés.

Para realizar una identificación del lenguaje, primero se tiene que crear un modelo de los lenguajes de interés. Como parte del sistema, Nexidia provee una aplicación que crea un modelo basado en muestras de audio de cada lenguaje que se desea identificar. En general, se necesitan al menos 20 horas de cada lenguaje para realizar el modelo.

Además de las muestras de audio, la creación de los modelos requiere el uso de uno o más discriminadores de lenguaje provisto por Nexidia. Los discriminadores proveen una línea base para comparar lenguajes durante el proceso de entrenamiento. El número de discriminadores que se usa aumenta el tiempo de ejecución y la exactitud del modelo.

Este software fue probado con el conjunto de evaluación del reconocimiento del lenguaje NIST 2005 [NIST 2005 LRE], el cual comprende 7 idiomas, Inglés, Hindú, Japonés, Coreano, Español y Tamil.

En adición al producto de Identificación del Lenguaje actualmente disponible comercialmente, Nexidia está trabajando en algunas áreas de investigación para mejorar la exactitud y la utilidad de éste. Las características actualmente bajo investigación son:

- Proveer la habilidad de agregar un nuevo lenguaje al modelo existente sin tener que reentrenar el modelo completo.
- Detección del dialecto.
- Proveer un mecanismo para determinar cuándo un lenguaje no está contemplado en el modelo.
- Detectar automáticamente las porciones de los clips de audio que contienen habla para mejorar la precisión ignorando el silencio, ruido o cualquier otra señal que no sea de habla.

3.1.2 Phonexia

El sistema Phonexia [Phonexia 2012] fue desarrollado por el grupo llamado Speecg@FIT de la facultad de Tecnología de la Información en Universidad de Brno, República Checa.

El sistema ayudará a distinguir el lenguaje hablado en una muestra de audio. Esta tecnología puede ser utilizada para dirigir llamadas a alguien en que hable el mismo lenguaje. El sistema es rápido y fácil de integrar con un centro de llamadas.

Casos de uso:

- Seguridad
- Análisis de tráfico en red.
- Centros de llamada.
- Agencias de noticias.
- Búsquedas web que trabajen con habla.

Este software realiza la identificación a través de 2 enfoques: Acústico y Fonético. El enfoque acústico se realiza convirtiendo la muestra a un espectro de frecuencia que es modelado por métodos estadísticos. El enfoque Fonético transcribe las muestras a cadenas de fonemas usando un reconocedor de fonemas altamente exacto y después modela las cadenas por modelos estadísticos del lenguaje.

Los lenguajes disponibles en el sistema son : Árabe, Inglés, Farsi, Francés, Alemán, Hindú, Japonés, Coreano, Mandarín, Español, Tamil y Vietnamita. Sin embargo, el software también permite agregar nuevos lenguajes a través de la entrada de muestras de entrenamiento grabadas en 8000Hz, 8 ó 16 bits de calidad y formato WAV o RAW.

La aplicación fue evaluada utilizando el conjunto de evaluación NIST 2003 [NIST 2003 LRE].

3.1.3 SLISSAL

El sistema SLISSAL [Peché 2009] fue desarrollado en el departamento de ingeniería eléctrica, electrónica y de computación de la Universidad de Pretoria.

El objetivo principal de este software es la identificación del lenguaje entre los 11 lenguajes oficiales en Sudáfrica.

Esta aplicación usa la arquitectura Reconocimiento de Fonemas en Paralelo seguido del Modelado del Lenguaje. En esta arquitectura, las muestras de audio son procesadas primero en una cadena de fonemas, antes de que la clasificación sea realizada. El primer paso extrae información fonotáctica en la forma de símbolos fonémicos de la señal de audio. La cadena resultante de símbolos es entonces pasada a algún tipo de modelo del lenguaje que es usado para determinar el lenguaje más probable del conjunto de lenguajes objetivo.

Una vez que la señal de audio es procesada por el reconocedor de fonemas, la cadena resultante es clasificada. Entonces, se regresa el lenguaje con la probabilidad más alta.

El sistema fue probado solamente con muestras de audio del corpus Lwazi, que consiste en audio telefónico así como transcripciones de los 11 lenguajes oficiales de Sudáfrica.

3.1.4 IRIT/SAMOVA

Este software aún se encuentra en desarrollo por el grupo SAMOVA [Samova 2011] del Instituto de Investigación en Informática de Toulouse. Su objetivo es desarrollar un sistema completo de Identificación Automática del Lenguaje usando la acústica, la fonotáctica y la prosodia.

Este sistema se compondrá de 3 módulos:

Modulo Acústico – Fonético
Proponen un enfoque basado en las siguientes afirmaciones

- Las tipologías de los lenguajes distinguen entre vocales y consonantes.
- Esas tipologías pueden ser usadas para identificar un lenguaje desde su descripción fonológica.
- Las características más importantes para caracterizar un sonido dependen de su clase fonética.
- Cuando se modelan sonidos homogéneos, es más fácil tomar en cuenta algunas restricciones específicas.

Estas afirmaciones nos llevan a considerar un modelo diferenciado para cada sistema fonológico redefinido con respecto a las restricciones ligadas a la representación acústica-fonética del habla espontánea.

Módulo Fonotáctico

El módulo fonotáctico está justo detrás el decodificador acústico-fonético, este usa las unidades acústicas-fonéticas dadas por el modelo fonotáctico diferenciado.

Modulo Prosódico

Desde un punto de vista prosódico, los lenguajes difieren en su ritmo y entonación. Su hipótesis es que el ritmo es producido por la periodicidad de un patrón que puede ser sílaba, que es una unidad específica del lenguaje. Se propone un algoritmo para la extracción automática e independiente del lenguaje.

3.2 Tipos de sistemas LID

Los sistemas LID están basados en las propiedades lingüísticas de los lenguajes, las cuales son extraídas de muestras de habla. El desempeño de cada sistema LID depende de la cantidad y exactitud de la información discriminativa incorporada en el sistema.

La mayoría de los sistemas LID cuentan con información espectral obtenida a través del análisis espectral de la señal de habla, tal como lo son las propiedades acústicas de las unidades del sonido y sus secuencias. En adición a la información espectral, algunos sistemas LID incorporan información prosódica. La Figura 6, muestra la arquitectura

general de un ejemplo de sistema LID basado en los diferentes enfoques de acercamiento [Decker 2009].

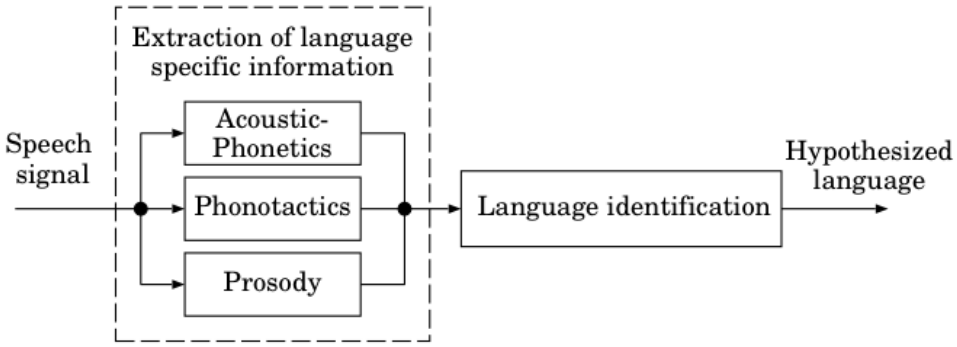

Figura 6 Arquitectura general de un sistema LID utilizando diferente información discriminativa.

A continuación se describen los tres enfoques más utilizados en el desarrollo de sistemas LID

3.2.1 Sistemas Acústicos

El sistema desarrollado en esta investigación es considerado puramente acústico ya que toma información directamente de la muestra del habla usando la transformada wavelet. Como veremos a continuación, el uso de la transformada wavelet para caracterizar la señal de habla no es habitual en los sistemas LID acústicos, por lo cual podemos considerar que el uso de esta herramienta matemática como algo novedoso.

Los sistemas LID puramente acústicos, tienen por objetivo capturar diferencias esenciales entre lenguajes, modelando distribuciones de características espectrales directamente. Esto se realiza típicamente extrayendo un conjunto de características espectrales que dependen del lenguaje a partir de segmentos de habla y usando un clasificador estadístico para identificar patrones dependientes del lenguaje en tales características.

La opción más usada de las características espectrales es la llamada “Coeficientes Cepstrales de Frecuencia Mel” (MFCC). Las características MFCC son calculadas durante

el análisis de muestras de habla siguiendo el siguiente proceso: la muestra de habla segmentada es transformada al dominio de la frecuencia basado en una transformada lineal de coseno de un espectro de potencia logarítmica en una escala Mel no lineal. Los 13 coeficientes cepstrales más bajos y sus primera y segunda derivaciones forman el vector de características cepstrales.

El principio fundamental en el modelado de la mayoría de los sistemas LID basado en información acústica – fonética es el mismo: un vector de características es generado de acuerdo a la función de densidad de probabilidad, la cual es escogida dependiendo de las especificaciones de la aplicación. La elección más usada en los sistemas LID son Modelos de Mezclas Gaussianas (GMM).

En los GMM la probabilidad es modelada como una suma ponderada de funciones de densidad normal multivariada (Gaussiana) con parámetros estimados en información de entrenamiento. El primer sistema GMM LID basado en características MFCC y una regla de decisión de máxima verosimilitud es el propuesto por [Zissman 1993]. Los GMM no son computacionalmente costosos y no requieren información fonotáctica etiquetada para la fase de entrenamiento. Sin embargo, los sistemas LID basados en GMM realizan una clasificación estática, en el sentido de que los vectores de características son asumidos como independientes unos de otros, de tal forma que las secuencias de vectores no son tomadas en cuenta.

Para superar esta desventaja y capturar la información discriminativa de lenguajes que reside en los patrones dinámicos de características espectrales, algunos sistemas LID han usado los llamados Modelos Ocultos de Markov (HMM). Los HMM son un conjunto de estados, cada uno de los cuales es asociado a una distribución de probabilidad. La estructura de estados de los HMM toma en consideración el comportamiento secuencial de los modelos del habla, mientras que la función de probabilidad captura patrones acústicos en el tiempo. El primer sistema LID basado en HMM fue propuesto por [House – Neuberg 1977].

Un intento por incorporar información de la información del habla fue realizado por [Torres Carrasquillo 2002b] quien propuso el uso de "Shifted Delta Cepstra" (SDC) como características espectrales para sistemas GMM. Incluso, Torres Carrasquillo combinó exitosamente características SDC con GMM de alto orden.

Otro tipo de acercamiento acústico al LID, usa Máquinas de Soporte Vectorial (SVM). La esencia de las SVM recae en la representación de los límites de separación entre lenguajes en un espacio multidimensional en términos de unos puntos cruciales obtenidos a partir del conjunto de entrenamiento, llamados vectores de soporte.

Actualmente, la mayoría de los sistemas LID basados en SVM, adoptan sus ideas de los sistemas basados en GMM. Estos pueden usar características MFCC y SDC, así como técnicas de compensación de la variabilidad del hablante.

3.2.2 Sistemas Fonotácticos

La fonotáctica, las limitaciones de las frecuencias relativas de las unidades de sonido y sus secuencias en el habla, es una de las fuentes de información más ampliamente usadas en el LID. Su popularidad se debe a su implementación simple y escalable, además de su relativo alto poder de discriminación entre lenguajes. El enfoque fonotáctico usa un marco probabilístico y se fundamenta en la propiedad inherente de cada lenguaje de exhibir frecuencias y dependencias específicas entre los fonemas individuales en una declaración hablada. Este fenómeno puede ser fácilmente ilustrado con un ejemplo: Mientras que la combinación del fonema /i/ seguido por /ç/ ocurre frecuentemente en el Alemán, ésta no existe en el Inglés.

La identificación automática fonotáctica es todavía un área activa de investigación con dos problemas principales:

1. La alta exactitud de los modelos estadísticos usualmente incrementa la complejidad de los modelos en términos del número de parámetros que tienen que ser estimados. Además, se requiere información de entrenamiento adicional.

2. La calidad de los sistemas fonotácticos es influenciada por la exactitud de reconocimiento de unidades de sonido del habla.

Los modelos probabilísticos fonotácticos pueden tener una variedad de estructuras, la más popular de estas es el modelo conocido como N-gramas, que consiste básicamente analizar una secuencia de elementos en subsecuencias de *n* elementos y la relación que hay entre ellos [Navratil 2006].

Una pregunta natural que surge al utilizar este enfoque es ¿Qué reportorio fonético debe ser usado para representar las declaraciones multilingüistas como una secuencia discreta de "tokens"? [Zissman - Singer 1993] usaron un solo reconocedor de fonemas del idioma inglés y probaron que el modelado fonotáctico en términos de fonemas de un lenguaje es factible para la identificación de otros lenguajes. Como se ilustra en la Figura 7, la muestra de entrada es decodificada en una secuencia de unidades discretas por medio de un reconocedor de fonemas, la cual es sometida a un análisis fonotáctico, por ejemplo el modelado por bi-gramas.

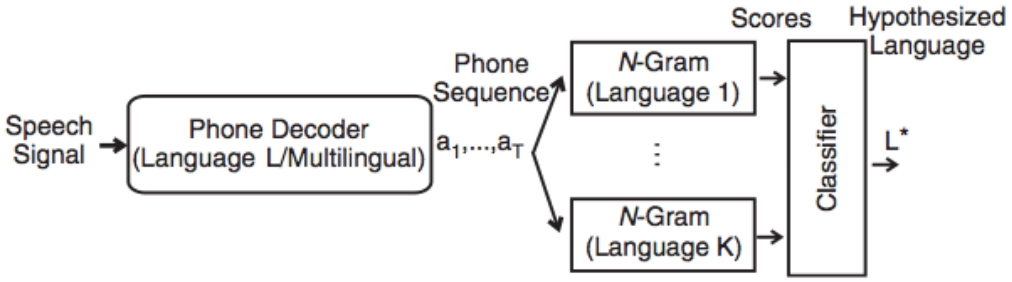

Figura 7 Arquitectura Fonotáctica con un solo Decodificador de Fonemas.

Para una mejor cobertura de fonemas entre lenguajes, [Hazen 1994] utiliza un decodificador único con un repertorio de fonemas multilingüistas y con un número variable de fonemas. [Yan – Barnard 1996] emplearon seis reconocedores dependientes del lenguaje de fonemas para representar mejor el inventario de fonemas e incrementar la robustez del clasificador. Esta arquitectura con decodificadores en paralelo también es utilizada por

[Zissman 1993] y es llamada PPRLM (Parallel Phone Recognition followed by Language Modeling). La Figura 8 muestra la arquitectura con múltiples decodificadores de fonemas.

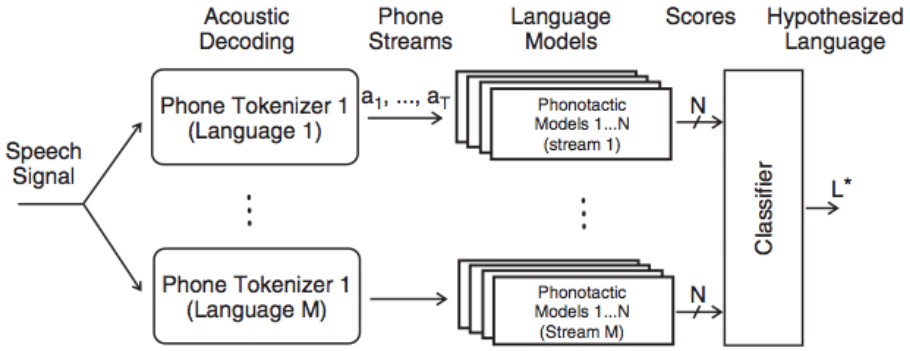

Figura 8 Arquitectura Fonotáctica con múltiples decodificadores de fonemas.

3.2.3 Sistemas Prosódicos

Además de la información acústica y fonotáctica, la información prosódica puede ser usada para discriminar lenguajes. A pesar de esto, los sistemas LID prosódicos son relativamente raros. La mayoría de este tipo de sistemas utiliza el tono, la entonación y la prominencia para caracterizar los lenguajes. Dichas características son extraídas de dos componentes de la señal: la frecuencia fundamental (*F0*) y la amplitud.

En [Eady, 1982], por ejemplo, se estudiaron dos lenguajes con diferentes propiedades prosódicas: el Inglés el cual pertenece a la categoría de lenguajes que marcan diferentes niveles de prominencia por medio de la *F*0 y la amplitud, y el idioma Chino, en el cual los patrones de tonos son lexicamente distintivos. El autor comparó el tiempo de contorno de la frecuencia fundamental y la energía extraída de oraciones del mismo tipo y encontró diferencias únicas específicas de cada lenguaje, tal como una mayor tasa de cambio en cada contorno y una fluctuación más alta entre sílabas individuales para el chino.

Sin embargo, el uso de la prosodia para identificar lenguajes conlleva ciertos problemas. La dificultad de obtener una evaluación clara de la utilidad de un componente prosódico para

LID deriva del gran número de factores adicionales que influencia a la F0 y la amplitud. Estos incluyen:

- Características específicas del hablante (tipo de voz, estado emocional, salud, etc.)
- Elección léxica (características de las palabras, tal como el estrés de la palabra y tono léxico).
- Contenido sintáctico de la muestra (declaración, pregunta).
- Contenido pragmático (contexto influye en la interpretación del significado).

En muchos casos, particularmente en los lenguajes que pertenecen a la misma familia (o del mismo tipo prosódico), las variaciones prosódicas de cada lenguaje son sobrescritas por otras más dominantes, dificultando su extracción e identificación.

3.3 Bases de Datos para la Evaluación

Hasta principios de los años 90, las investigaciones LID sufrían de una carencia de corpus multilingüistas de habla y de dominio público para evaluar y comparar los sistemas, así como sus modelos y sus enfoques. Aunque algunos países de la Unión Europea están activamente involucrados en lo esfuerzos para colectar recursos multilingüistas, la mayoría de las colecciones coordinadas dedicadas a la investigación LID vienen principalmente de los EE. UU. Las bases de datos más conocidas y usadas en la investigación LID son las siguientes:

- OGI_TS 11L y 22L
- CallHome 6L
- CallFriend 12L
- SpeechDat
- GlobalPhone
- Fisher
- Mixer

La mayor parte de la experimentación realizada en esta investigación se realizó sobre la bases de datos OGI_TS 11L y Lwazi, las cuales son descritas a continuación.

3.3.1 OGI_TS

La base de datos OGI_TS fue creada en [Yeshwant Muthusamy et al. 1992] para la investigación científica en el Centro para el Entendimiento de Lengua Hablada del Instituto de Graduados de Oregon (OGI).

Para la creación de la base de datos se promovió la realización de llamadas telefónicas bajo el lema "Dona tu voz para la ciencia" en algunas universidades y grupos de noticias de computación de los Estados Unidos. Además, se realizó una publicación describiendo el proyecto de investigación y la necesidad de voluntarios en la radio y periódicos.

La recolección de las muestras se realizó usando un Gradient Technology Desklab conectado a una computadora Sun 4/110 por un puerto SCSI. El dispositivo se programó para responder las llamadas, reproducir archivos digitales en cada uno de los 10 lenguajes solicitando muestras de habla y digitalizar las repuestas para un periodo de tiempo. Las muestras fueron grabadas a 8000 KHz y una resolución de 16 bits.

El protocolo de grabación fue diseñado para obtener muestras de:

- vocabulario fijo
    - ¿Cuál es tu lengua materna?
    - ¿Qué lengua habla la mayor parte del tiempo?
    - Por favor recite los siete días de la semana.
    - Por favor, diga los número del 1 al 10.
- breves descripciones de tema específico
    - Hable acerca del clima en su ciudad.
    - Hable acerca de algo que le guste de su ciudad natal.

    - Describa su más reciente comida.
    - Describa la habitación desde la que está llamando.
- tema libre.

De esta base de datos tomaron 50 muestras de audio para cada uno de los idiomas seleccionados, haciendo un total de 450 muestras.

3.3.2 Lwazi

La meta del proyecto [Lwazi 2011] es proveer a los ciudadanos del Sudáfrica con información de servicios en sus lenguas maternas, a través de líneas telefónicas de una forma eficiente y costeable. Éste fue comisionado por el Departamento de Arte y Cultura de la República de Sudáfrica

Una de las tareas de este proyecto, es la creación de una córpora para poder realizar pruebas en las diferentes áreas de investigación que el proyecto abarca hasta la fecha, las cuales son: Text-to-Speech (TTS) y Automatic Speech Recognition (ASR).

La base de datos Lwazi abarca los 11 idiomas de Sudáfrica (SA), los cuales son:

- IsiZulu e IsiXhosa, lenguajes Nguni, los dos lenguajes más hablados en SA. Juntos, estos dos forman la lengua materna del 41% de la población de SA
- Los tres lenguajes Sotho: Setswana, Sepedi, y Sesotho. Juntos forma el 26% de las lenguas maternas de la población de SA
- Los dos lenguajes Nguni menos hablados en SA: Siswati e IsiNdebele.
- Xitsonga y Tshivenda, la lengua materna del 2% y 4% de la población en SA
- Los otros dos lenguajes oficiales en SA son los lenguajes Germánicos: Inglés y Afrikaans

Para todos estos lenguajes, se desarrollaron diccionarios electrónicos de pronunciación y bases de datos de texto y habla. La base de datos para ASR consiste de aproximadamente

200 hablantes por lenguaje produciendo habla libre y leída, grabada sobre la línea telefónica. Cada hablante produjo aproximadamente 20 muestras cortas de habla, 16 de estas fueron seleccionadas aleatoriamente de una base de datos fonéticamente balanceada y el resto consistió de palabras cortas y frases: respuestas a preguntas abiertas, respuestas a preguntas si/no, fechas y números. La muestra de hablantes fue seleccionada para proveer un perfil balanceado tomando en cuenta la edad, el género y el tipo de teléfono (fijo o móvil).

Dado que la base de datos generada por el proyecto Lwazi no estaba enfocada hacia la investigación LID, ésta tuvo que ser adaptada para poder ser utilizada en nuestro proyecto. La adaptación realizada consistió en seleccionar de cada lenguaje solo aquellas muestras que tuvieran una duración mayor a los 10s, esto debido a que nuestro sistema solo procesa muestras de 10s. Al realizar la selección de las muestras se tuvieron que excluir de las pruebas los lenguajes Inglés y Africaans, ya que ninguna de sus muestras tenía una duración mayor a los 10s requeridos.

3.4 Estudios sobre clasificación – Identificación

Durante el desarrollo de esta investigación se marcó la diferencia entre clasificar e Identificar. Mientras que en la clasificación se utiliza un solo conjunto de muestra tanto para entrenar y generar los modelos como para evaluar dichos modelos, en la identificación se utilizan dos conjuntos muestras, el conjunto de entrenamiento y el conjunto de evaluación para dichas tareas. Esto nos permite tener un escenario más parecido al final, en el que se realizará la identificación del idioma sobre muestras completamente fuera del conjunto de entrenamiento y de los modelos.

En seguida se muestras los resultados obtenidos para los enfoques descritos anteriormente.

La tabla 1 describe las condiciones y los resultados obtenidos por los sistemas LID con un enfoque puramente acústico, este enfoque es el mismo utilizado en esta investigación. La

mayoría de sistemas considerados en la tabla 1 utilizan la base de datos OGI_TS, la cual no cuenta con un conjunto de entrenamiento, por lo cual podemos suponer que realizan una clasificación de idiomas y no una identificación.

Tabla 1 Algunos sistemas acústicos LID y sus tasas de desempeño**.**

Sistema	Tarea	Duración de la prueba	Desempeño	Referencia
GMM (40 comp.)	OGI-10L	10s/45s	50%53%	[Zissman, 1996]
GMM (16 comp.)	OGI-11L	10s/45s	49%/53%	[Hazen y Zue, 1997]
HMM-fonema	OGI-10L	10s	59.7%	[Lamel y Gauvian, 1994]
NN-Silábico	OGI-10L	10s	55%	[Li, 1994]
GMM-Vocálico	5L (habla leída)	21s	70%	[Farinas et al., 2002]

El enfoque fonotáctico es el enfoque más popular e implementado en los sistemas LID. Este enfoque es el que mejor resultados ha reportado. La tabla 2 hace un resumen de algunos sistemas LID desarrollados. Cabe destacar que los sistemas LID desarrollados por Torres Carrasquillo utilizan la base de datos CallFriend, la cual sí incluye un conjunto de evaluación. Por lo cual podemos suponer que estos sistemas sí hacen una identificación del lenguaje.

Tabla 2 Ejemplos de sistemas LID fonotácticos y sus tasas de reconocimiento.

Sistema	Tarea	Duracion de la señal de prueba	Tasa de identificación	Referencia
Tokenizer multilingüe de trigramas interpolados	OGI-11L	10s/45s	62.7%/77.5%	[Hazen y Zue, 1997]
Bigramas (1 tokenizer) dependientes del género	OGI-10L	10s/45s	54%/72%	[Zissman, 1996]
PPPRLM (3 tokenizers)	OGI-10L	10s/45s	63%/79%	[Zissman, 1996]
PPRLM (6 tokenizers)	OGI-6L	10s/45s	74%/84.8%	[Yan and Barnard, 1995]
N-gramas extendidos (PPRLM con 6 flujos)	OGI-6L	10s/45s	86.4%/97.5%	[Navrátil 2001]
Modelado de flujos	OGI-10L	Mezclados	65%	[Parandekar y Kirchoff,

cruzados.				2003]
GMM-Tokenizer	CallFriend (12L)	30s	63.7%	[Torres Carrasquillo et al, 2002]
GMM-Tok+PPRLM	CallFriend (12L)	30s	83%	[Torres-Carrasquillo et al, 2002]

Por último, la tabla 3 muestra los resultados obtenidos por sistemas que hacen uso del enfoque prosódico. Este enfoque es el que peores resultados presenta, quizás esta sea la razón por la cual es el enfoque menos usado, lo que hace que sea muy raro encontrar sistemas que usen solamente un este enfoque.

Tabla 3 Algunos componentes prosódicos y sus tasas de reconocimiento.

Sistema	Tarea	Duración de prueba	Tasa de identificación	Referencia
Duración	OGI-11L	10s/45s	31.7%/44.4%	[Hazen y Zue, 1997]
Ritmo	5L (habla leída)	21s	22%	[Farinas et al, 2002]
Características F0	OGI-2L (promediado)	45s	70%	[Rouas et al, 2003]

CAPÍTULO 4

METODOLOGÍA

El prototipo de software desarrollado en este proyecto hace uso de una metodología de Identificación de Lenguas. Esta metodología se basa en las investigaciones de clasificación y multiclasificación desarrolladas en [Reyes-Herrera 2007, Vargas-Martínez 2008, Medina – Trejo 2011 y Hernández – Zepeda 2011], con el propósito de realizar una identificación real de idiomas. La metodología puede ser dividida en dos secciones, la primera (Figura 9), enfocada a la elaboración de los modelos matemáticos (biclasificadores) y la segunda (Figura 10), enfocada en la multiclasificación de idiomas utilizando los biclasificadores producto de la primera sección.

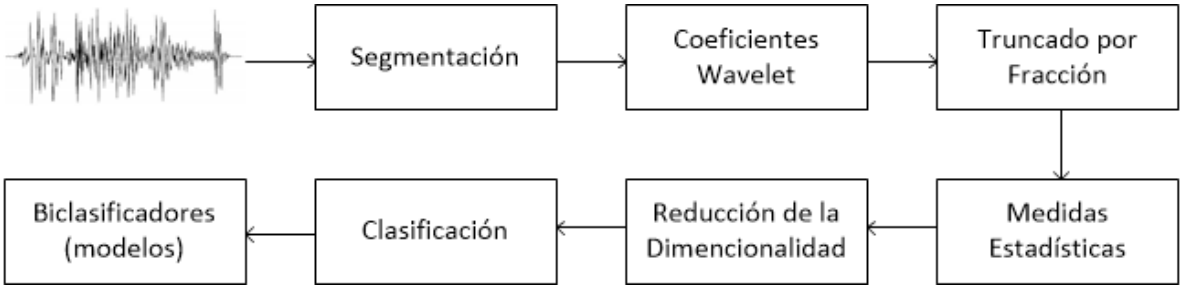

Figura 9 Metodología para la Generación de los Modelos Matemáticos

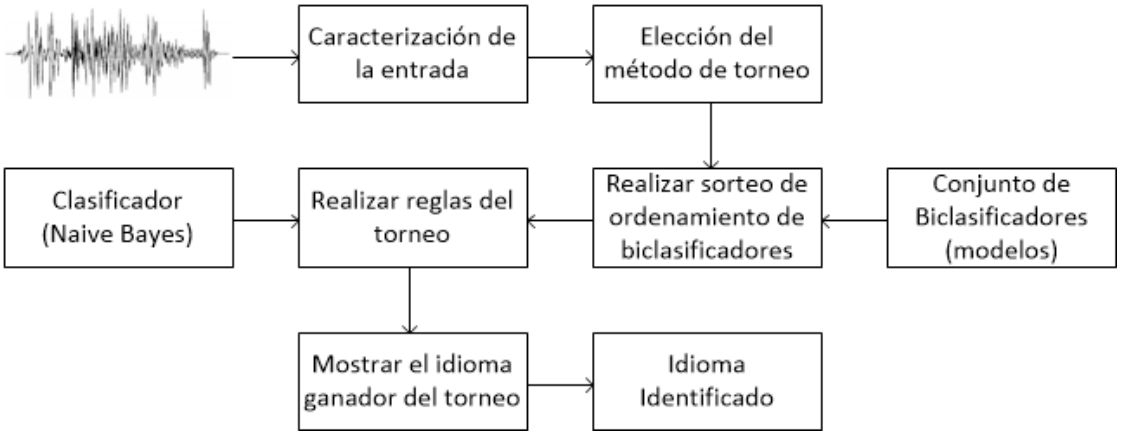

Figura 10 Metodología de Multiclasificación

La metodología encargada de elaborar los modelos fue desarrolla inicialmente por [Reyes-Herrera 2007], quien utilizó por primera vez la transformada wavelet para caracterizar las muestras de audio. Después, [Vargas-Martínez 2008] agregó el uso de medidas estadísticas para representar la señal, a diferencia de Reyes Herrera quien utilizaba solamente los coeficientes wavelet. Por último, las investigaciones realizadas por [Medina-Trejo 2011, Hernández Zepeda 2011] desarrollaron la multiclasificación de idiomas mediante la realización de torneos entre los biclasificadores y agregaron dos medidas estadísticas más.

A continuación se describen las fases que comprenden la metodología que genera los modelos.

Segmentación

La primera parte del procesamiento, comprende la segmentación de la señal de audio en pequeñas señales de 1 segundo. En nuestro caso, lo que se desea es una mejor representación de la señal en términos del ritmo con menos datos. En este sentido, el uso de los wavelets logra una alta resolución en bajas frecuencias (donde se encuentra el ritmo) con periodos cortos de tiempo. El número de segmentos será igual al número de segundos que dure la señal de habla.

Coeficientes Wavelet

En este punto, se toma la señal segmentada y se le aplica la trasformada wavelet Db2, la cual utiliza cuatro coeficientes filtro para descomponer dicha señal, por lo tanto se aplicarán las transformaciones a todos los segmentos de un segundo de la señal. La trasformada es representada finalmente con los coeficientes wavelet producidas por la transformación.

Truncado por Fracción

El tipo de truncado que se utiliza es llamado truncado por fracción, pues se conserva una fracción establecida del conjunto original de datos inalterado, mientras los demás se establecen a cero. De los coeficientes wavelet que resultan del cálculo, los coeficientes de mayor magnitud representan las bajas frecuencias y los de menor magnitud las altas frecuencias [Timoshenko 2011]. Por tal razón, usamos un truncado con una fracción del

1%, esto nos permite usar solo el 1% de los coeficientes de mayor magnitud, una cantidad ya probada en el trabajo de [Reyes-Herrera et al. 2006], que se cree representa el ritmo, y que logra que dichos coeficientes aporten el mayor peso de información para la siguiente parte del proceso.

Medidas Estadísticas

Inicialmente la metodología solo utilizaba los coeficientes wavelet para caracterizar la señal, sin embargo, esto generó una nueva problemática, el manejo de la gran cantidad de datos. Una manera de reducir la gran cantidad de datos, y a la vez describirlos, es representarlos mediante medidas estadísticas que capturen el contenido original de estos. Las medidas utilizadas en este trabajo fueron: la media, desviación estándar, máximo y mínimo, así como las medidas estadísticas skewness y kurtosis, agregadas recientemente a la metodología [Medina-Trejo 2011, Hernández Zepeda 2011] .

Reducción de la Dimensionalidad

Una vez que las características estadísticas han sido extraídas, se construyen los clasificadores o modelos matemáticos en formato weka, a pesar de la reducción mediante variables estadísticas, las instancias son muy grandes, y se procede a aplicar ganancia de información para seleccionar a los atributos más significativos en cada clasificador, esta técnica funciona como filtro, el umbral elegido para filtrar fue de cero, lo que indica que aquellos que tengan ganancia de información cero o debajo de cero serán eliminados.

Clasificador Mediante Validación Cruzada

La validación cruzada es un método utilizado en estadística, el cual divide en subconjuntos iguales, a un conjunto de datos, donde cada subconjunto es analizado o clasificado (en nuestro caso) con el resto, lo que permite analizar un conjunto completo sin tener que dividirlo y solo para la validación cruzada usando Naive Bayes fue de 10 subconjuntos (folds).

4.1 Adaptación de la Metodología que Genera los Modelos a MATLAB y Java

Si bien la implementación de la metodología hasta antes de este trabajo se encontraba funcional y estable, fue necesario realizar algunas adaptaciones para desarrollar un sistema que realizara una identificación real del idioma.

La mayor parte de la implementación de los trabajos antecedentes giraba alrededor del software acústico Praat. Este software realizaba las tareas de lectura de la señal, segmentación, extracción de los coeficientes wavelet, y el truncado por fracción. Sin embargo, este software no es programable, de modo que pueda ser parte de un sistema final. Debido a esto, fue necesario implementar dichas tareas en una herramienta programable, para lo que se utilizó MATLAB. A continuación se describe la implementación realizada.

Segmentación

La segmentación consiste simplemente en dividir la señal de entrada en pequeños segmentos de un segundo para obtener una mejor representación de la señal en términos del ritmo con menos datos.

La lectura de las muestras de audio se realiza usando la función de MATLAB *wavread* de la siguiente manera

[senial,frecuencia__de_muestreo]=wavread(“direccion_del_archivo”);

almacenando la muestra de audio en el vector *senal* con la longitud del número de segundos por la frecuencia de muestreo y obteniendo la frecuencia de muestreo de la señal en la variable *frecuencia_de_muestreo.*

Una vez que se tiene la señal almacenada se procede a realizar la segmentación con la siguiente instrucción desarrollada

*senial_segmentada=senial(k*frecuencia_muestreo+1 : (k+1)*frecuencia_muestreo);*

Básicamente, lo que se realiza con esta instrucción es acceder a las posiciones del vector que corresponde al *k* segundo que se quiere obtener usando la información obtenida en la sección anterior.

Coeficientes Wavelet

Para la extracción de los coeficientes se usó la siguiente instrucción predefinida por MATLAB

[C,L]=wavedec(senial_segmentada,nivel_descomposicion,'db2');

la cual, dados el tipo de wavelet, el nivel de descomposición y la señal, regresa en el vector C los coeficientes wavelet de la señal y en L las posiciones correspondientes a cada nivel de descomposición. La siguiente figura muestra la estructura de los vectores resultantes de la descomposición de la señal a través de MATLAB.

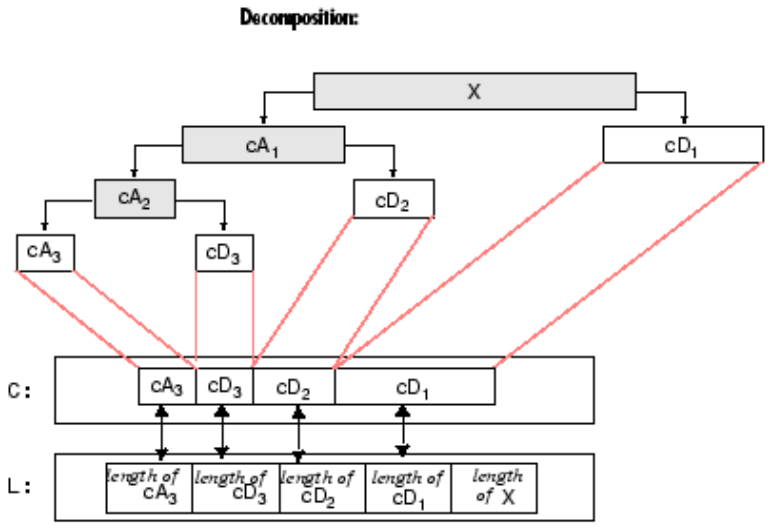

Figura 11 Descomposición por Niveles de la Transformada Wavelet

Truncado por Fracción

A diferencia de Praat, en MATLAB no existe una instrucción de alto nivel que realice el truncado por fracción, por lo cual, se tuvo que realizar la implementación de dicho truncado.

El código desarrollado para realizar el truncado por fracción es el siguiente

```
porcentaje=0.01;
[C,L]=wavedec(senial_segmentada,nivel_descomposicion,'db2');
numero_coeficientes=L(1);
[wavelets_ordenados, indices_originales] = sort(abs(C(1:numero_coeficientes)),'descend');
wavelets_truncados=zeros(numero_coeficientes,1);
for q=1:numero_coeficientes*porcentaje
  wavelets_truncados(indices_originales(q))=C(indices_originales(q));
end
```

Figura 12 Truncado por fracción en MATLAB

Interfaz de Usuario

Otra adaptación que se realizó en esta sección fue el desarrollar una interfaz gráfica que permitiera la generación de los modelos de una manera sencilla para el usuario final.

Para realizar esta actividad se utilizaron las siguientes tecnologías y librerías

-Java 1.6
-Commons – io – 2.4
-MatlabControl 4.0

La librería Commons IO de Apache es una librería que contiene clases de utilidades para asistir el desarrollo de funcionalidades como flujo de archivos, filtrado de archivos, comparadores, entre otras. En nuestra implementación nos permitió realizar tareas como filtrar archivos por extensión durante la selección de muestras y copiar de archivos al entorno local. De esta librería se utilizó la versión 2.4.

MatlabControl es otra librería utilizada en nuestro sistema. Esta librería nos permite invocar funciones de Matlab desde programas desarrollados en Java ejecutados en una máquina virtual distinta. La prueba de concepto de esta idea fue desarrollada inicialmente por Kamin

Whitehouse de la universidad de Virginia. Posteriormente, esta idea fue continuada y desarrollada por Joshua Kaplan de la Universidad Brown, quien además de mejorar el trabajo de Kamin, desarrolló un RMI (Java Remote Method Invocation) para poder llamar métodos de Matlab desde Java en máquinas virtuales diferentes.

El uso de MatlabControl nos permitió realizar las actividades administrativas de nuestro software desde Java y realizar la parte de procesamiento matemático desde Matlab a través de llamadas a sus funciones. Las actividades quedaron separadas como se muestra en la siguiente figura.

Figura 13 Actividades Realizadas por cada Tecnología

La parte administrativa del sistema es manipulada por el usuario a través de una interfaz gráfica de Java, la siguiente figura muestra dicha interfaz.

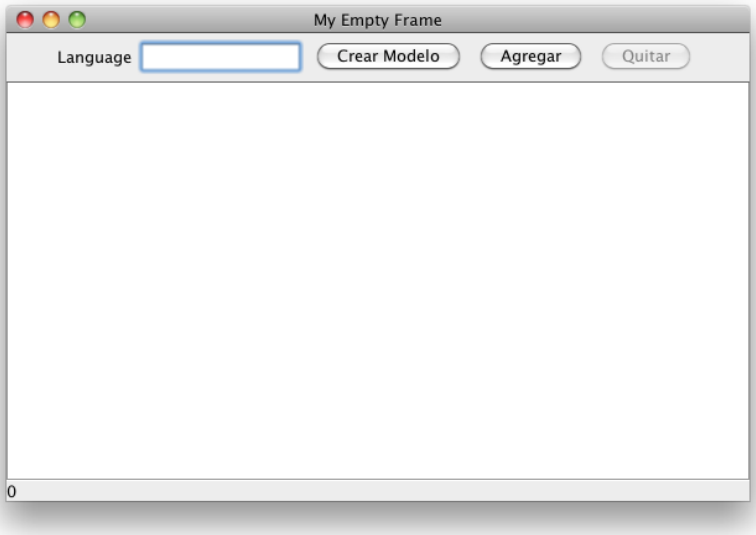

Figura 14 Interfaz Gráfica del Sistema para la Generación de Modelos

Esta interfaz se introduce el nombre del idioma y se agregan las 50 muestras de audio. Después, éstas son copiadas al directorio del sistema y se les aplican las fases de la metodología. La Figura 15 muestra la estructura interna de archivos del sistema.

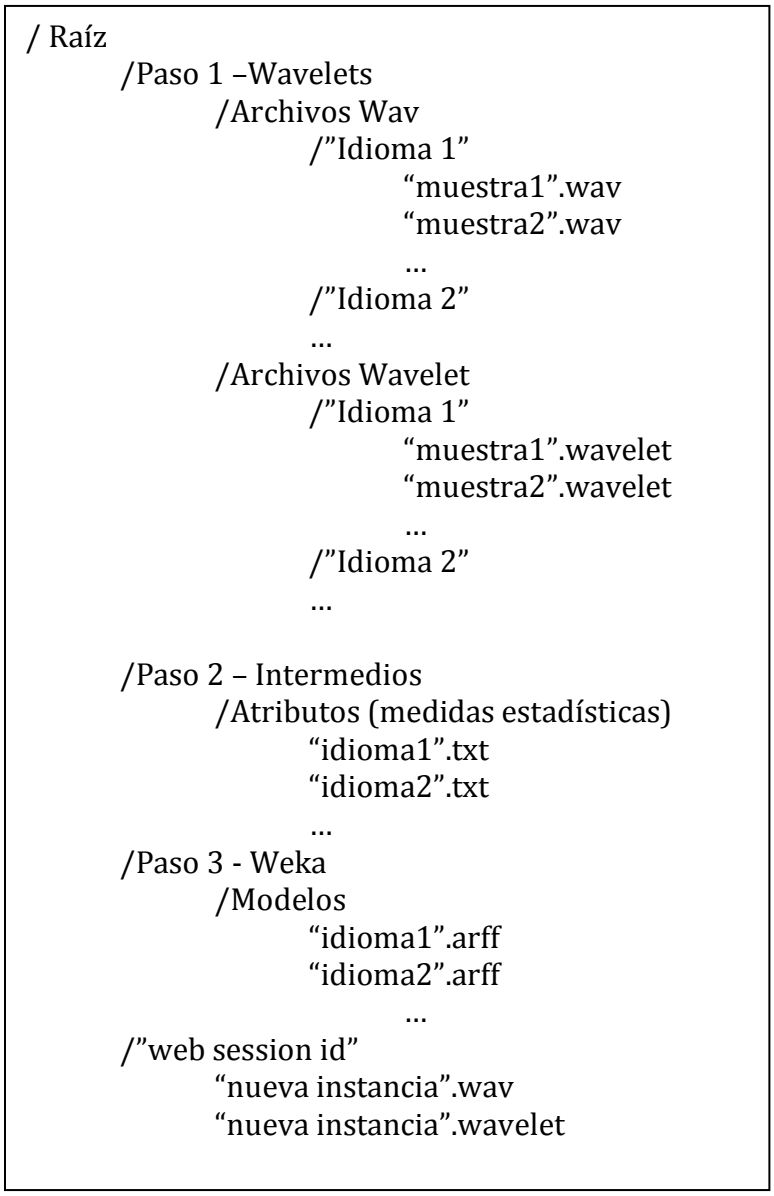

Figura 15 Estructura Interna de Archivos del Sistema

La salida del esta parte del sistema son modelos, lo cuales fueron evaluados mediante la validación cruzada que se ha usado en los trabajos antecedentes.

El desarrollo realizado hasta esta parte comprende la generación de modelos matemáticos que representan a cada idioma o lenguaje.

4.2 Adaptación de la Metodología de Multiclasificación

Además de modificar la implementación de la metodología que genera los modelos, también fue necesario adaptar la multiclasificación generada en [Medina – Trejo 2011].

La identificación del idioma se realiza utilizando los biclasificadores obtenidos con la primera parte del software. En la identificación todos los idiomas soportados por el sistema, es decir, los idiomas de los cuales se tiene modelos matemáticos, compiten en un torneo (Round Robin) para determinar que lenguaje es el que ha sido asignado más veces a la nueva instancia.

El trabajo de [Medina – Trejo 2011] generó una metodología para realizar la multicalsificación de idiomas haciendo uso de los biclasificadores. Sin embargo, dicha metodología contempla una variante por grupos del torneo Round Robin. En nuestro caso, por las características de nuestra investigación se implementó la multiclasificación para un solo grupo.

La implementación de la multiclasificación contempla un solo grupo en el que todos los competidores compiten entre sí, de modo que el competidor con más victorias es el idioma identificado.

La siguiente figura muestra la interacción entre las tecnologías usadas para realizar la identificación de idiomas.

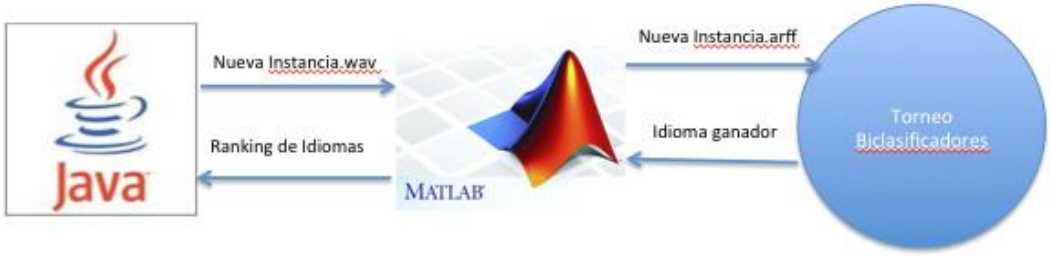

Figura 16 Interacción entre las tecnologías para desarrollar la identificación.

Una vez que el torneo es finalizado, el servidor manda el ranking de los idiomas al cliente para mostrar el resultado de que idioma fue el ganador.

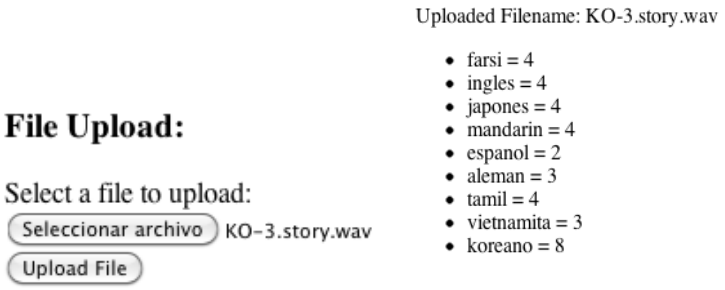

Figura 17 Lado Cliente del Sistema.

Captura de una Nueva Instancia

Para la caracterización de las instancias a identificar, solo se tuvieron que modificar los códigos de Matlab que se encargan de caracterizar las 50 muestras para que caractericen una única instancia. También, se adaptó el programa que genera el archivo *.arff de las 50 instancias de modo que genere el archivo correspondiente para una nueva instancia.

Una vez que se contaba con el programa necesario para caracterizar una sola instancia, se prosiguió a desarrollar el software necesario para la recepción de la nueva instancia a través de internet desde una página web o desde un dispositivo móvil.

El desarrollo del servidor web encargado de recibir las nuevas instancias se realizó usando Java como lenguaje de programación.

El servidor consiste básicamente de una página web en la que se carga la nueva muestra de audio y un servlet[3] en el que se validan las características del archivo. Dentro de este mismo servlet se realiza la conexión con Matlab para la caracterización de la nueva

[3] Clase de Java que es usada para extender las capacidades de los servidores que almacenan aplicaciones accedidas por medio del modelo de programación solicitud – respuesta [Oracle 2010].

instancia. La siguiente figura muestra la estructura general del proceso de identificación de una nueva instancia.

Figura 18 Comunicación entre el Cliente y el Servidor

CAPÍTULO 5

EXPERIMENTACIÓN Y RESULTADOS

Durante el desarrollo de este proyecto se realizaron una gran variedad de pruebas con distintos objetivos. Entre dicho objetivos se encuentran el determinar la validez de los coeficientes wavelet obtenidos por MATLAB, determinar la mejor configuración del software para clasificar e identificar idiomas, así como determinar el número de idiomas soportados que mejora el promedio de clasificación del sistema.

La experimentación realizada en esta investigación se desarrolló completamente en el siguiente entorno:

Software
S. O. Mac OS X Snow Leopard
Matlab 2011b para Mac
Java 1.6

Hardware
Macbook White
Intel Core 2 Duo 2.2 GHz
4GB Memoria Ram
120 GB D. D.

Los objetivos de estas pruebas, así como sus resultados obtenidos, son detallados a continuación.

5.1 Validación de los Wavelets de Matlab

Uno de los objetivos específicos de este trabajo fue realizar una implementación de la metodología usando MATLAB como principal herramienta.

Inicialmente, se pretendía replicar el algoritmo que usa Praat para la extracción de los coeficientes wavelet, sin embargo, no se encontró información acerca del algoritmo que usa dicha herramienta para realizar los cálculos. Por lo tanto, se optó por usar las funciones predefinidas de MATLAB, las cuales obtienen el mismo resultado que las funciones implementadas por [Lonut Danaila 2006].

Wolfram Mathematica [Mathematica 8] y Maple [Maple 14] fueron las dos herramientas matemáticas usadas para realizar la comparación de los wavelets obtenidos por Praat y MATLAB. De esta comparación pudimos observar que ninguno de los programas producía los mismos coeficientes. Sin embargo, al graficar los coeficientes obtenidos por MATLAB Y Wolfram Mathematica, se observó una gran similitud entre las gráficas. La siguiente figura muestra las gráficas de los coeficientes obtenidos por ambos programas.

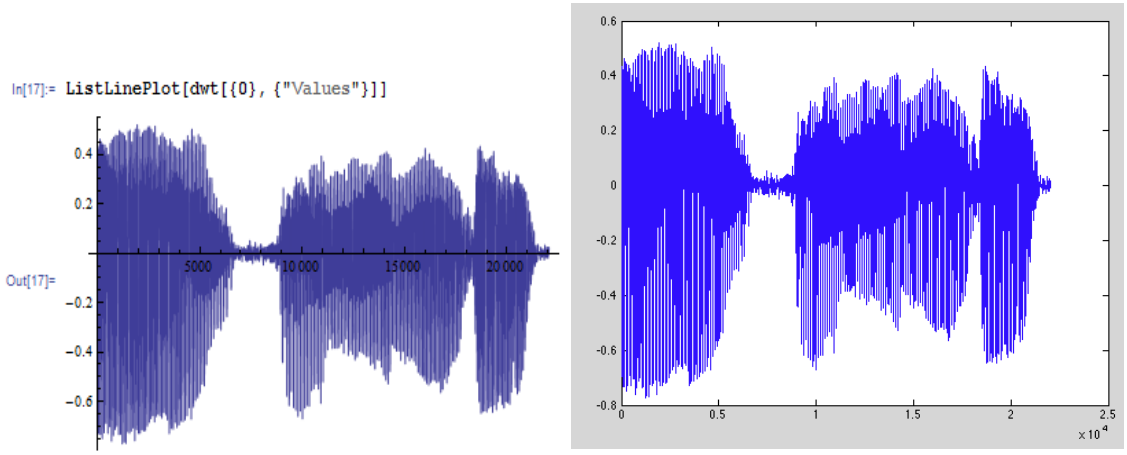

Figura 19 Gráfica de los coeficientes wavelests obtenidos por MATLAB(izq.) y Mathematica (der.)

Después de desarrollar la mayor parte de la metodología en MATLAB, la siguiente opción que se consideró para probar la validez de los coeficientes resultantes de MATLAB, fue realizar los modelos biclasificadores y evaluarlos con la siguiente configuración:

- Muestras de 10s con frecuencia de muestreo de 8kHz
- Un nivel de descomposición de la transformada wavelet
- Seis medidas estadísticas, máximo, mínimo, media, mediana, skewness y kurtosis
- Truncado por fracción de 1%
- Precisión de 9 decimales.
- Clasificador Naive Bayes
- Validación cruzada con 10 folds

La tabla 4 muestra los porcentajes de clasificación obtenidos por nuestra implementación para dicha configuración.

Tabla 4 Porcentaje de clasificación para la prueba de 10s, 1 nivel de descomposición y 6 medidas estadísticas, 98.88%.

	Español	**Farsi**	**Japonés**	**Coreano**	**Mandarín**	**Tamil**	**Vietnamita**	**Inglés**
Alemán	98	97	100	99	97	98	99	99
Español		98	100	100	97	100	100	98
Farsi			100	98	99	100	100	99
Japonés				100	98	98	100	99
Coreano					99	99	99	98
Mandarín						99	99	99
Tamil							99	100
Vietnamita								99

Al comparar nuestros resultados con los mejores obtenidos en los trabajos antecedentes [Hernández – Zepeda 2011] para la misma configuración, podemos observar que los coeficientes wavelet de MATLAB obtienen mejores resultados, al tener un promedio de 98.88% de clasificación, mientras que los otros obtienen un promedio de 95.64%. La tabla 5 muestra los resultados obtenidos en [Hernández – Zepeda 2011].

Tabla 5 Porcentaje de clasificación obtenido por [Hernández – Zepeda 2011]

	Español	**Farsi**	**Japonés**	**Coreano**	**Mandarín**	**Tamil**	**Vietnamita**	**Inglés**
Alemán	93	95	95	95	94	99	98	95
Español		99	95	95	95	96	97	99
Farsi			97	97	93	94	100	96
Japonés				97	96	91	95	94
Coreano					98	97	98	93
Mandarín						93	95	92
Tamil							95	94
Vietnamita								97

Los resultados obtenidos nos muestran un buen porcentaje de clasificación de los modelos creados a partir de los coeficientes wavelets obtenidos por MATLAB, sin embargo, se decidió realizar más pruebas para explorar otras configuraciones en la generación de los modelos.

5.2 Experimentación sobre Clasificación.

Una vez que se validó el buen desempeño de los modelos generados por MATLAB, se procedió a experimentar con nuevas configuraciones de parámetros en busca de mejorar el promedio de clasificación de los modelos.

El primer parámetro que se modificó fue el nivel de descomposición de la transformada wavelet. La prueba en este sentido fue generar los modelos de clasificación con nivel de descomposición 2. Las siguientes tablas muestran los resultados obtenidos.

Tabla 6 Porcentaje de clasificación para la prueba de 10s, 2 niveles de descomposición y 6 medidas estadísticas, 97.38%.

	Español	**Farsi**	**Japonés**	**Coreano**	**Mandarín**	**Tamil**	**Vietnamita**	**Inglés**
Alemán	94	99	98	97	100	97	95	94
Español		97	96	96	98	98	100	96
Farsi			98	98	99	100	99	98
Japonés				99	99	97	97	99
Coreano					97	98	97	97
Mandarín						96	97	97
Tamil							99	98
Vietnamita								94

Tabla 7 Porcentaje de clasificación para la prueba de 10s, 2 nivel de descomposición y 4 medidas estadísticas, 96%.

	Español	**Farsi**	**Japonés**	**Coreano**	**Mandarín**	**Tamil**	**Vietnamita**	**Inglés**
Alemán	93	98	96	98	100	98	96	90
Español		92	98	97	94	100	95	95
Farsi			98	96	98	98	96	94
Japonés				97	97	97	99	95
Coreano					92	99	95	96
Mandarín						95	96	95
Tamil							96	96
Vietnamita								91

Los resultados obtenidos por los modelos generados a partir de los coeficientes obtenidos de la transformada wavelet con un nivel de descomposición dos son buenos, sin embargo, no superan a los obtenidos por la transformada con solo un nivel de descomposición.

Las siguientes pruebas consistieron en aumentar en el nivel de precisión en las variables de MATLAB. La precisión de estas variables fue aumentada de 9 a 15 decimales. Las siguientes tablas muestran los resultados obtenidos para estas pruebas.

Tabla 8 Porcentaje de clasificación para la prueba de 10s, 1 nivel de descomposición, 6 medidas estadísticas y 15 decimales de precisión, 98.77%.

	Español	Farsi	Japonés	Coreano	Mandarín	Tamil	Vietnamita	Inglés
Alemán	98	96	99	99	98	98	99	99
Español		98	100	100	97	99	100	98
Farsi			100	98	98	100	99	99
Japonés				100	99	98	100	98
Coreano					99	98	99	98
Mandarín						97	98	99
Tamil							99	100
Vietnamita								99

Tabla 9 Porcentaje de clasificación para la prueba de 10s, 2 niveles de descomposición, 6 medidas estadísticas y 15 decimales de precisión, 97.41%.

	Español	Farsi	Japonés	Coreano	Mandarín	Tamil	Vietnamita	Inglés
Alemán	94	99	98	97	100	97	95	94
Español		97	96	96	98	98	99	96
Farsi			98	98	99	98	100	98
Japonés				99	99	97	99	99
Coreano					92	98	97	97
Mandarín						96	97	97
Tamil							98	98
Vietnamita								94

De los resultados anteriores podemos observar que la prueba con sólo un nivel de descomposición y 15 decimales de precisión se obtiene un promedio de clasificación de 98.77777778 muy cercano al 98.88888889% que obtiene la misma prueba pero con sólo 9 decimales.

Otra característica que se probó fue cambiar la longitud de las muestras. Las pruebas realizadas hasta este punto se han realizado tomando solamente 10s de las muestras. En esta prueba se consideró tomar 50s de cada una las muestras de audio, sin embargo, el archivo GE-185.story.wav del idioma Alemán tiene una longitud de 44s. Esto nos llevó a tomar

solamente 44s de 50s de cada muestra. La siguiente tabla muestras los resultados obtenidos para las prueba con 44s.

Tabla 10 Porcentaje de clasificación para la prueba de 44s, 2 niveles de descomposición, 6 medidas estadísticas y 9 decimales de precisión, 96.35%.

	Español	**Farsi**	**Japonés**	**Coreano**	**Mandarín**	**Tamil**	**Vietnamita**	**Inglés**
Alemán	97	95	99	99	97	97	96	92
Español		97	97	99	96	95	94	91
Farsi			100	99	98	97	96	99
Japonés				98	100	93	99	91
Coreano					99	88	97	98
Mandarín						96	97	87
Tamil							92	99
Vietnamita								90

Por último, se decidió realizar una prueba en la que las muestras se obtuvieran de una base de datos diferente. Las pruebas realizadas hasta este punto hacen uso de la base de datos OGI TS. Para esta prueba se utilizó la base de datos llamada Lwazi [Lwazi 2011] que consiste en muestras de audio y transcripciones de 11 lenguas o idiomas de Sudáfrica. De las muestras usadas, solo se tomaron 10s para la generación de los modelos con un nivel de descomposición, 6 medidas estadísticas y 9 decimales de precisión. La siguiente tabla muestras los resultados obtenidos.

Tabla 11 Porcentaje de clasificación para la prueba de 10s, 1nivel de descomposición, 6 medidas estadísticas y 9 decimales de precisión, 98.02%.

	isixhosa	**isizulu**	**sepedi**	**sesotho**	**setswana**	**siswati**	**tshivenda**	**xitsonga**
isindebele	99	95	100	100	100	97	99	99
isixhosa		99	98	100	96	94	100	97
isizulu			100	97	97	96	99	99
sepedi				99	98	98	100	100
sesotho					97	99	99	96
setswana						95	97	96
siswati							98	97
tshivenda								99

De todas las pruebas realizadas se concluyó que las mejores condiciones para generar los modelos de clasificación son las siguientes:

- Un nivel de descomposición.
- 6 medidas estadísticas.
- Longitud de 10s de las muestras de audio.
- Precisión de 9 decimales.

Si bien los modelos obtienen excelentes resultados para estas pruebas, estas no representan el escenario real en el cual serán usados, ya que en una aplicación real, las instancias que serán identificadas no son contenidas en los modelos.

5.3 Experimentación sobre Identificación

La experimentación mostrada en esta sección se enfocó a realizar pruebas que evaluaran los modelos en un escenario más parecido al real, a aquel escenario en el que los modelos serían usados por la aplicación desarrollada.

La diferencia que hacemos entre clasificación e identificación comprende básicamente el hecho de que la clasificación toma muestras, que de alguna manera, están incluidas en el modelo, mientras que la identificación toma muestras que no se encuentran en el modelo.

La evaluación realizada en esta sección utilizó la base de datos OGI TS, de la cual se formaron dos conjuntos de muestras, el conjunto de entrenamiento y el conjunto de evaluación. Como ya se ha mencionado anteriormente la base de datos con la que se cuenta, consta de 50 muestras de 44s para 9 lenguajes. De estas muestras, el conjunto de entrenamiento estuvo formado por los primeros 10s de cada muestra, mientras que el conjunto de evaluación estuvo formado por los últimos 10s de cada muestra.

Las pruebas realizadas en esta evaluación consistieron en introducir al sistema cada una de las muestras del conjunto de evaluación para realizar la identificación del lenguaje de dicha muestra, de tal modo que teniendo 10 muestras por idioma en el conjunto de evaluación, un

50% de acierto significa que en 5 de 10 muestras de un idioma, dicho idioma fue identificadas correctamente. La identificación se realiza a través de un torneo, en este caso Round Robin, en el que el que cada idioma compite por ser el idioma identificado. Al final de cada torneo el idioma que haya sido identificado más veces será el ganador. Para estas pruebas se consideran ganadores a los lenguajes que se encuentran en el 1er o 2º lugar del ranking de idiomas arrojado por el sistema.

La siguiente tabla muestra los resultados obtenidos por la primera prueba con la configuración que obtuvo los mejores resultados de clasificación en la sección anterior.

Tabla 12 Prueba con truncado por fracción de 1%, porcentaje de acierto 37.77%

	Porcentaje de Acierto
Alemán	50
Español	30
Farsi	20
Japonés	30
Coreano	60
Mandarín	60
Tamil	10
Vietnamita	10
Inglés	70

Los resultados obtenidos por esta primera prueba nos permitieron observar que las pruebas realizadas hasta este punto no realizaban una identificación real del idioma, sino que solamente realizaban una clasificación de idiomas entre muestras contenidas en los modelos.

Al obtener resultados de identificación bajos, se optó por continuar con la experimentación de parámetros. Las pruebas realizadas en la sección anterior exploran opciones tales como el nivel de descomposición, el número de muestras, la precisión y el número de medidas

estadísticas, dejándonos como única opción a explorar el porcentaje de truncado por fracción.

Los valores a explorar de porcentaje de truncado por fracción fueron 3, 5, 8 y 10 por ciento. En seguida se muestran las tablas correspondientes para cada prueba.

Tabla 13 Porcentajes de Identificación obtenidos para la experimentación con el truncado por fracción.

	1%	**3%**	**5%**	**8%**	**10%**
Alemán	50	20	40	50	50
Español	30	50	60	50	50
Farsi	20	30	30	20	20
Japonés	30	10	40	20	20
Coreano	60	60	90	10	10
Mandarín	60	60	10	40	40
Tamil	10	10	10	0	0
Vietnamita	10	20	80	40	40
Inglés	70	70	30	30	30
Promedio	37.77	35.5	43.33	28.88	34.44

De la experimentación realizada observamos que la configuración con un truncado por fracción de 5% obtiene los mejores resultados de identificación.

Además de realizar experimentación con el truncado por fracción también se experimentó con el tamaño de los grupos.

Las pruebas anteriores comprendían un grupo de 9 idiomas, por lo tanto, se decidió realizar experimentación reduciendo el tamaño del grupo. Los tamaños a experimentar serían 5 y 3.

Las pruebas para grupos de 5 idiomas consistieron de tres grupos para cada porcentaje de truncado por fracción, al igual que la prueba de 9 idiomas se experimentó con los porcentajes de truncado 1, 3, 5 y 8. Para cada porcentaje se realizaron 3 pruebas, cada una con un grupo de 5 idiomas determinados por el ranking de posiciones obtenido para el

mismo porcentaje de truncado en la prueba de 9 idiomas. Por ejemplo, para la prueba con 9 idiomas con un trucado por fracción de 5% se obtiene el siguiente ranking.

1. Coreano 90%
2. Vietnamita 80%
3. Español 60%
4. Alemán 40%
5. Japonés 40%
6. Farsi 30%
7. Inglés 30%
8. Tamil 10%
9. Mandarín 10%

De este ranking se toman los idiomas posicionados en los primeros lugares para formar un grupo, otro grupo se forma con los idiomas en las últimas posiciones, y por último se forma un grupo en el que se incluya a los 3 idiomas en los primeros lugares y 2 en los últimos lugares. Los grupos para el ranking mostrado quedarían como sigue.

5 Primeros Lugares	5 Últimos Lugares	3 Primeros y 2 Últimos lugares
• Coreano	• Japonés	• Coreano
• Vietnamita	• Farsi	• Vietnamita
• Español	• Inglés	• Español
• Alemán	• Tamil	• Tamil
• Japonés	• Mandarín	• Mandarín

Los idiomas para las pruebas con grupos de 3 idiomas también estuvieron determinados por el ranking de la prueba de 9 idiomas correspondiente. Para determinar los idiomas en cada grupo el ranking es partido en 3 grupos de igual tamaño, de modo que los idiomas quedan agrupados de acuerdo a su posición.

Por ejemplo, para el ranking mostrado anteriormente los grupos de idiomas para la prueba con un truncado por fracción de 5% quedarían como sigue

Posiciones 1, 2 y 3	Posiciones 4, 5 y 6	Posiciones 7, 8 y 9
• Coreano	• Alemán	• Inglés
• Vietnamita	• Japonés	• Tamil
• Español	• Farsi	• Mandarín

Al igual que en las pruebas con nueve idiomas, el conjunto de evaluación consistió de 10 muestras tomadas aleatoriamente de la base de datos OGI TS, de las cuales se toman los últimos 10 segundos de cada muestra, mientras que el conjunto de entrenamiento toma los primero 10 segundos de las 50 muestras, evitando así que dichos conjuntos contengan muestras idénticas.

En la experimentación con solo 3 idiomas se considera un acierto en la identificación cuando el idioma queda en el primer lugar de la clasificación, mientras que para las pruebas de 5 idiomas se considera acierto quedar en el primer o segundo lugar de la clasificación, al igual que en la prueba de 9 idiomas.

A continuación se presentan los resultados obtenidos para los porcentajes 1, 3, 5 y 8 de truncado por fracción para los grupos de 5 idiomas.

Tabla 14 Porcentaje de Acierto de las pruebas con 5 idiomas para los porcentajes 1, 3, 5 y 8 de truncado por

	1%	3%	5%	8%
5 primeros	44	40	38	50
5 últimos	50	40	48	42
3 primeros, 2 últimos	48	48	44	50

De los resultados obtenidos para las pruebas con grupos de 5 idiomas podemos observar que la configuración con truncado por fracción de 8% obtiene los mejores resultados al

tener un promedio de 5.0 para de 2 de sus 3 grupos. En seguida se muestra la tabla con los idiomas participantes en esta prueba.

Tabla 15 Porcentaje de Acierto para la Prueba con 8% de Truncado por Fracción

Grupo de Idiomas	Promedio de Acierto
Alemán, Español, Mandarín, Inglés, Vietnamita	50
Farsi, Japonés, Coreano, Tamil, Inglés	42
Alemán, Español, Farsi, Japonés, Coreano	50

Además de realizar pruebas con grupos de 5 idiomas, también se realizaron pruebas con grupos de 3 idiomas. En seguida se muestra la tabla con el resumen de los resultados obtenidos en estas pruebas.

Tabla 16 Porcentaje de Acierto de las pruebas con grupos de 3 idiomas para los porcentajes 1, 3, 5 y 8 de

	1%	3%	5%	8%
1er, 2º y 3er lugar	33	50	43	40
4º, 5º y 6º lugar	36	26	40	33
7º, 8º, 9º lugar	40	36	26	20

Los mejores resultados son los obtenidos por la configuración con un truncado por fracción de 3%, obteniendo un 5.0 para uno de sus 3 grupos.

Tabla 17 Porcentaje de Acierto para la Prueba con 3% de Truncado por Fracción

Idiomas	Promedio de Acierto
Inglés, Mandarín, Coreano	50
Alemán, Farsi, Español	26
Japonés, Vietnamita, Tamil	36

Al analizar los resultados que arrojan estas dos pruebas, así como la anterior con 9 idiomas, podemos notar que realizar la identificación con grupos de idiomas más pequeños solamente arroja una pequeña mejora.

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

El objetivo principal de este proyecto fue el de desarrollar un software para la identificación automática del lenguaje. Además de desarrollar el código necesario, es imprescindible realizar pruebas a dicho software para garantizar el rendimiento y la estabilidad esperada. Estas pruebas se presentaron en el capítulo anterior y consistieron en la variación de parámetros como el nivel de descomposición de la transformada wavelet, el número de medidas estadísticas e incluso el número de idiomas contemplados en el sistema. Estas y otras pruebas nos llevaron a las siguientes conclusiones.

6.1 Conclusiones

La primera conclusión debido a la gran distancia de resultados entre los trabajos antecedentes y el presente trabajo, es la diferencia entre clasificar e identificar. Mientras que los trabajos antecedentes se dedicaron a clasificar instancias dentro del mismo modelo, este trabajo se enfocó al problema real de identificar el lenguaje de nuevas instancias.

En cuanto a la experimentación presentada en el capítulo anterior, esta nos permitió llegar a las siguientes conclusiones.

- La herramienta Matlab obtiene coeficientes wavelet distintos en magnitud (normalizados de manera distinta) a los obtenidos por el software Praat pero los modelos matemáticos generados con dichos coeficientes obtienen mejores promedios de clasificación que los reportados en los trabajos antecedentes.
- Al probar con dos bases de datos distintas, Lwazi y OGI TS, podemos concluir que el tipo de idioma no afecta la clasificación siempre y cuando las muestras de audio compartan las mismas características técnicas.
- La mejor configuración de parámetros para realizar la clasificación de idiomas es la siguiente

    - Un nivel de descomposición en la transformada wavelet db2.
    - 6 Medidas Estadísticas.
    - Longitud de 10s de las muestras de audio.
    - Precisión de 9 decimales.
- En cuanto a la identificación, la mejor configuración del software es la siguiente
    - Un nivel de descomposición.
    - 6 Medidas Estadísticas.
    - Longitud de 10s de las muestras de audio.
    - Precisión de 9 decimales.
    - 5% de truncado por fracción
- Los bajos resultados de identificación nos llevaron a modificar el tamaño del grupo de idiomas concursantes probando grupos de 3, 5 y 9. De esta experimentación se concluyó que el número de idiomas participantes no afecta significativamente el promedio de identificación.

La conclusión final obtenida a partir de la experimentación realizada, aunado a lo observado en el estado del arte, es que el enfoque acústico por si solo no es suficientemente bueno como para obtener promedios de identificación elevados y que es necesario incorporar otro tipo de información u enfoque, como lo es el enfoque fonotáctico.

6.2 Trabajo Futuro

Lo anterior nos llevan a presentar como trabajo futuro la posibilidad de desarrollar un sistema LID que utilice el enfoque fonotáctico y que además, sea capaz de integrarse con el sistema desarrollado en este proyecto, siguiendo la tendencia de algunos investigadores a desarrollar sistemas LID que integran ambos enfoques.

También se propone como extensión del sistema actual, el desarrollo de las aplicaciones para dispositivos móviles modernos como Android, iOS, RIM y Windows Phone, con la finalidad de realizar la grabación de las nuevas instancias directamente desde estos dispositivos.

Por último se sugiere realizar nueva experimentación utilizando bases de datos con muestras de audio de características superiores a las observadas en las bases de datos Lwazi y OGI TS.

En paralelo a este trabajo, se está generando una nueva base de datos de voces indígenas de México en el proyecto "Corpus de Lenguas Indígenas Mexicanas para la Identificación Automática del Lenguaje Hablado" por el Ing. Carlos A. Hernández Zepeda.

6.3 APORTACIONES

A continuación se listan las aportaciones de este trabajo

- Primer prototipo funcional de un sistema LID acústico.
- Interfaz gráfica para añadir nuevos idiomas.
- Implementación de la metodología de clasificación en MATLAB con mejores resultados.
- Experimentación para determinar la mejor configuración de parámetro para clasificar e identificar idiomas.

Referencias

[Abercrombie 1967] D. Abercrombie. Elements of general phonetics. Edinburgh University Press, Edin- burgh, 1967.

[Apache IO] Apache Commons, http://commons.apache.org/io/

[Battle 1987] G. Battle. A block spin construction of ondelettes. Part I: Lemari´e functions. Com- mun. Math. Phys., 110:601–615, 1987.

[Barnwell 1986] M. J. T. Smith and T. P. Barnwell III. Exact reconstruction for tree-structured subband coders. IEEE Trans. Acoust., Speech, and Signal Proc., 34(3):431–441, June 1986.

[Briggs 1987] W. L. Briggs. A Multigrid Tutorial. SIAM, Philadelphia, 1987.

[Burt 1987] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. IEEE Trans. Commun., 31(4):532–540, April 1983.

[Cavaretta 1991] A. S. Cavaretta, W. Dahmen, and C. Micchelli. Stationary subdivision. Mem. Amer. Math. Soc., 93:1–186, 1991.

[Croisier 1976] A. Croisier, D. Esteban, and C. Galand. Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques. In Int. Conf. on Inform. Sciences and Systems, pages 443–446, Patras, Greece, August 1976.

[Chui 1997] Chui, C. K., Wavelets: A Mathematical Tool for Signal Processing, SIAM, Filadelfia, 1997

[Daubechies 1998] I. Daubechies. Orthonormal bases of compactly supported wavelets. Commun. on Pure and Appl. Math., 41:909–996, November 1988.

[Dyn 1990] N. Dyn and D. Levin. Interpolating subdivision schemes for the generation of curves and surfaces. In W. Haussmann and K. Jetter, editors, Multivariate Approximation and Interpolation, pages 91–106. Birkauser Verlag, Basel, 1990.

[Farinas et al. 2002] Farinas, J., Pellegrino, F., Rouas, J.-L., Andre-Obrecht, R. (2002). Merging segmental and rhythmic features for automatic language identification. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Orlando, FL.

[Gibbs 1899] Gibbs J., Fourier's series, letter in Nature, 59 (1898), p. 200.

[Goupillaud - Morlet 1984] P. Goupillaud, A. Grossman, and J. Morlet. Cycle-octave and related transforms in seismic signal analysis. Geoexploration, 23:85–102, 1984/85. Elsevier Science Pub.

[Haar 1910] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. Math. Annal., 69:331– 371, 1910.

[Hernández – Zepeda 2011] Hernández Zepeda Carlos Arturo, Identificación Automática de Lenguas, Utilizando la familia de Wavelets Dbn, Medidas Estadísticas y Minería de Datos, Tesis de Licenciatura, Instituto Tecnológico de Ciudad Madero, 2011

[House – Neuberg A. S. House and E. P. Neuburg. Toward automatic identification of

1977] the language of an utterance. i. preliminary methodological considerations. In Proc. of the Journal of Acoustic Society of America, volume 62(3), pages 708–713, September 1977.

[Hazen 1994] Hazen, T., Zue, V. (1994). Recent improvements in an approach to segment- based automatic language identification. In: *Proceedings of the Interna- tional Conference on Spoken Language Processing*. Yokohama, Japan, pp. 1883–1886.

[Hazen – Zue 1997] Hazen, T., Zue, V. (1997). Segment-based automatic language identification. *Journal of the Acoustical Society of America* 101 (4), 2323–2331.

[Java 1.6] Java 1.6, http://www.java.com/es/download/faq/java_6.xml

[Kohavi 1995] Ron Kohavi, *A Study of Crros – Validation and Bootstrap for Accuracy Estimation and Model Selection*, International Joint Conference on A. I. 1995

[Ladefoged 1975] P. Ladefoged. A Course in Phonetics. Harcourt Brace Jovanovich Inc., New York, 1975.

[Lamel-Gauvain 1994] Lamel, L., Gauvain, J. (1994). Language identification using phone-based acous- tic likelihoods. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Adelaide, Australia.

[Lemarie 1988] P. G. Lemari´e. Ondelettes a` localisation exponentielle. J. Math. pures et appl., 67:227–236, 1988.

[Li 1994] Li, K. (1994). Automatic language identification using syllabic spectral features. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Adelaide, Australia, pp. 297–300.

[Lonut Danaila 2006] Ionut Danaila et al, An Introduction to Scientific Computing, 2006.

[Lwazi 2011] Badenhorst, J., Van Heerden C., Davel M., and Barnard Etienne, Collecting and evaluating speech recognition corpora for 11 South African languages (2011).

[Maple 14] Maple Soft, Technical Coputing software for Engineers.

[Mathematica 8] Wolfram Mathematica 8, Technical Computing software.

[MATLAB R2011a] MathWorks, MATLAB, The language of Technica Computing, 2011.

[MatlabControl] MatlabControl, https://code.google.com/p/matlabcontrol/

[Mallat 1989] S. Mallat. Multiresolution approximations and wavelet orthonormal bases of L2(R). Trans. Amer. Math. Soc., 315:69–87, September 1989.

[Medina – Trejo 2011] Medina Trejo César, Un Método para la Multiclasificación de Idiomas Usando la Transformada Wavelet, Tesis de Maestría, Instituto Tecnológico de Ciudad Madero, 2011.

[Meyer 1990] Y. Meyer. Ondelettes et Op´erateurs. Hermann, Paris, 1990. In two volumes.

[Mintzer 1985] F. Mintzer. Filters for distortion-free two-band multirate filter banks. IEEE Trans. Acoust., Speech, and Signal Proc., 33(3):626–630, June 1985.

[Morlet – Grossmann 1989] A. Grossmann and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. SIAM Journ. of Math. Anal., 15(4):723–736, July 1984.

[Navratil 2001] Navrátil, J. (2001). Spoken language recognition—a step towards multilinguality in speech processing. *IEEE Transactions on Speech and Audio Processing*. 9 (6), 678–685.

[Navratil 2006] Navrátil J., (2006). *Multilingual Speech Processing*, Phonotactic Modeling (251-261), San Diego, California USA: Elsevier.

[Nexidia 2012] Nexidia Inc, Automatic Language Identification of Audio, White Paper, USA. 2012.

[Oracle 2010] Oracle, "The Java EE 5 Tutorial", http://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html 2010

[Parendekar – Kirchoff 2003] Parandekar, S., Kirchhoff, K. (2003). Multi-stream language identification using data-driven dependency selection. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Hong Kong, China.

[Rosenfeld 1984] A. Rosenfeld, editor. Multiresolution Techniques in Computer Vision. Springer- Verlag, New York, 1984.

[Rouas 2003] Rouas, J.-L., Farinas, J., Pellegrino, F., Andre-Obrecht, R. (2003). Modeling prosody for language identification on read and spontaneous speech. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Hong Kong, China.

[Samova 2011] IRIT/SAMOVA, Language Identification System, White Paper, Toulouse Francia, 2009

[Smith 1984] M. J. T. Smith and T. P. Barnwell III. A procedure for designing exact reconstruction filter banks for tree structured sub-band coders. In Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc., San Diego, CA, March 1984.

[Timoshenko 2011] Ekaterina Timoshenko, Rhythm Information for Automated Spoken Language Identification, Tesis Doctoral, Universidad Técnica de Munich, 2011

[Torres Carrasquillo 2002a] Torres-Carrasquillo, P., Reynolds, D., Deller, J. Jr. (2002). Language identifi- cation using Gaussian mixture model tokenization. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Orlando, FL.

[Torres-Carrasquillo 2002a] Torres-Carrasquillo, P., Reynolds, D., Deller, J. Jr. (2002a). Language identifi- cation using Gaussian mixture model tokenization. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Orlando, FL.

[Tamara 2000] Tamara, V.,Compresión de señales empleando Wavelet, Tesis de pregrado, Departamento de Matemáticas. Universidad de los Andes, Santafé de Bogotá, 2000

[Torres Carrasquillo 2002b] P. Torres-Carrasquillo, E. Singer, M.A. Kohler, R. Green, D.A. Reynolds, and J.R. Deller Jr. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In

Proc. IEEE Int. Conf. on Spoken Language Processing (ICSLP), pages 719–722, Denver, Colorado, September 2002.

[Papoulis 1962] A. Papoulis. The Fourier Integral and its Applications. McGraw-Hill, New York, 1962.

[Peché 2009] Peché M., Davel M. H., y Barnard E., Development of a spoken language Identification for south African Languages, South African Institute of electrical Engineers, 2009.

[Pike 1945] K. L. Pike. The Intonation of American English. University of Migigan Press, 1945.

[Phonexia 2012] Phonexia, Phonexia Language Identification, White Paper, Brno República Checa 2012.

[Reyes-Herrera 2007] Reyes Herrera Ana Lilia, Un Método para la Identificación Automática de Lenguaje Hablado Basado en Características Suprasegméntales, Tesis doctoral, Instituto Nacional de Astrofísica Óptica y Electrónica, 2007.

[Vetterli 2011] M. Vetterli, J. Kovacevic and V. K. Goyal, *Foundations of Signal Processing*, 2011.

[Vetterli 2008] P. Prandoni, M. Vetterli, *Signal Processing for Communications* 2008

[Vetterli 1995] M. Vetterli, J. Kovacevic, *Wavelets and Subband Coding* 1995

[Witten 2005] Witten Ian, Frank Eibe, *Data Minig, Practical Machine Learning Tools and Techniques*, Second Edition, 2005

[Yan – Barnard 1995] Yan, Y., Barnard, E. (1995). An approach to automatic language identification based on language-dependent phone recognition. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Detroit, MI.

[Xia 2005] Y. Xia, W. Liu, and L. Guthrie, "Email Categorization with Tournament Methods," Proc. Int'l Conf. Application of Natural Language (NLDB), 2005.

[Yan – Barnard 1996] Yan, Y., Barnard, E., Cole, R. (1996). Development of an approach to automatic language identification based on phone recognition. *Computer Speech and Language* 10 (1), 37–54.

[Yeshwant Muthusamy et al. 1992] Muthusamy, Y., Cole, R., Oshika, B. (1992). The OGI multi-language telephone speech corpus. In: Proceedings of the International Conference on Spoken Language Processing. Banff, Alberta, Canada.

[Zissman - Singer 1993] Zissman, M., Singer, E. (1994). Automatic language identification of telephone speech messages using phoneme recognition and *N*-gram modeling. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Adelaide, Australia, pp. 305–308.

[Zissman 1993] Zissman, M. (1993). Automatic language identification using Gaussian mixture and hidden Markov models. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. pp. 399–402.

[Zissman 1996] Zissman, M. (1996). Comparison of four approaches to automatic language identification. *IEEE Transactions on Speech and Audio Processing*. 4 (1), 31–44.

ANEXO A

CÓDIGO PARA MODELAR MUESTRAS DE AUDIO DE IDIOMAS EN EL ENTORNO MATLAB.

1. Código para la extracción de los coeficientes wavelet.

```
function principal
clear all;
Numero_muestras=45;
Numero_segundos=10;
archivo_idiomas=fopen('Paso 1 - wavelets/newlanguages.txt','r');
porcentaje=0.08;
idiomasCell = textscan(archivo_idiomas,'%s','Delimiter','\n');
fclose(archivo_idiomas);
idiomas_CellString=idiomasCell{1}
Numero_idiomas=length(idiomas_CellString)
nivel_descomposicion=1;
for i=1:Numero_idiomas
   directorio_muestras=strcat(char(idiomas_CellString(i)),'.txt');
   archivo_muestras_idioma=fopen(directorio_muestras);
   muestras_cell=textscan(archivo_muestras_idioma,'%s',Numero_muestras);
    muestras_CellString=muestras_cell{:};
   for j=1:Numero_muestras
      direccion_muestra=['wav_files/' char(idiomas_CellString(i)) '/'
char(muestras_CellString(j)) ];
      [senial,frecuencia_muestreo]=wavread(direccion_muestra);
      temp=char(muestras_CellString(j));
      temp=temp(1:end-4);
      for k=0:Numero_segundos-1
         senial_segmentada=senial(k*frecuencia_muestreo+1:(k+1)*frecuencia_muestreo);
         [C,L]=wavedec(senial_segmentada,nivel_descomposicion,'db2');
         numero_coeficientes=L(1);
                  [wavelets_ordenados, indices_originales] =
         sort(abs(C(1:numero_coeficientes)),'descend'); % se
         wavelets_truncados=zeros(numero_coeficientes,1);
                  for q=1:numero_coeficientes*porcentaje
             wavelets_truncados(indices_originales(q))=C(indices_originales(q));
         end
         directorio_wavelets=['wavelet_files/' char(idiomas_CellString(i)) '/' temp
         int2str(k+1) '.wavelet'];
```

```
        archivo_escritura=fopen(directorio_wavelets,'w');
        fprintf(archivo_escritura,'%.9f \n',wavelets_truncados);
        fclose(archivo_escritura);

      end
   end
end
assignin('base', 'muestras_CellString', muestras_CellString);
assignin('base', 'idiomas_CellString', idiomas_CellString);
disp('Donne it!');
```

2. Código para obtener las medidas estadisticas de los coeficientes wavelet

```
function atributos
muestras_CellString= evalin ('base','muestras_CellString')
idiomas_CellString= evalin ('base','idiomas_CellString')
N_coef=4001;
newLanguage=char(idiomas_CellString{1})
NameFileAtri=['Atributos/' newLanguage '.txt'];
ruta=['wavelet_files/' newLanguage '/'];
Numero_elem=45;
Numero_segun=10;
fil=fopen(NameFileAtri,'w+');
m=[];
for i=1:Numero_elem
   nombrear = char(muestras_CellString{i});
   nombrear=nombrear(1:end-4);
   for s=1:Numero_segun
      rn=[ruta nombrear int2str(s) '.wavelet']
      cd ..
      cd 'Paso 1 - wavelets'
      pwd
      archivo=fopen(rn,'r');
      cd ..
      cd 'Paso 2 - intermedios'
      c=fscanf(archivo,'%f');
      m=[m c];
      fclose(archivo);
   end
   clear('C')
   clear('c');
   m=m';
   media=mean(m);
```

```
    desviacion=std(m);
    maximo=max(m);
    minimo=min(m);
    curtosis=kurtosis(m);
    obliquidade=skewness(m);
    atri=[media desviacion maximo minimo curtosis obliquidade];
    for j=1:length(atri)
        fprintf(fil,'%.9f ',atri(j));
    end
    fprintf(fil,'\n');
    clear('media');clear('desviacion');clear('maximo');clear('minimo');
    clear('curtosis');
    clear('obliquidade');
    clear('m');
    m=[];
    disp(i);
end
fclose(fil)
clear all;
```

3. Código para generar los archivos de las medidas estadicscticas en formato WEKA.

```
function cabecera
global file idio ii jj

newLanguage_CellString= evalin ('base','idiomas_CellString')
languages_CellString= evalin ('base','languages_CellString')
atributos=char('Media','Desviacion','Maximo','Minimo','Curtosis','Obliquidade');
a=cellstr(atributos);
clear('atributos');
primero='@RELATION lenguajes';
%file=fopen('cabecera.txt','w+');
fprintf(file,'%s\n',primero);
for i=1:6 %
    for j=1:4001
        linea=['@ATTRIBUTE' ' ' a{i,:} int2str(j) ' ' 'REAL'];
        fprintf(file,'%s\n',linea);
    end
    fprintf(file,'\n');
end
linea=['@ATTRIBUTE class            {' char(newLanguage_CellString{1}) ','
char(languages_CellString{jj}) '}'];
fprintf(file,'%s\n',linea);
linea='@DATA';
fprintf(file,'%s\n\n',linea);
```

```
%fclose(file);


function combinar
global C file idio ii jj tam1
cd ..
newLanguage_CellString= evalin ('base','idiomas_CellString')
archivo_languages=fopen('Paso 1 - wavelets/languages.txt','r');%se abre el archivo de
nombres de idioma
languagesCell = textscan(archivo_languages,'%s','Delimiter','\n');
languages_CellString=languagesCell{1}%extraccion del contenido de la cell obsteniendo
un cellstring
assignin('base', 'languages_CellString', languages_CellString);
Numero_idiomas=length(languages_CellString);
ext='.arff';
contador=1;
if Numero_idiomas ~ 0
   disp('Generando archivos arff..');
   for i=1:1
         archivo1=['Paso 2 - intermedios/Atributos/' char(newLanguage_CellString{1})
         '.txt'];
      A=load(archivo1);
      cd 'Paso 3 - weka';
      tam1=size(A,1);
      for j=1:Numero_idiomas
         archivo2=['Paso 2 - intermedios/Atributos/' char(languages_CellString{j}) '.txt'];
         cd ..
         B=load(archivo2);
         cd 'Paso 3 - weka';
         C=[A;B];
         clear('B')
         volcado=['weka/' char(newLanguage_CellString{1}) '-'
         char(languages_CellString{j}) ext];
         file=fopen(volcado,'w+');
         ii=i;jj=j;
         cabecera();
         fprintmat(C, file, idio, ii, jj, tam1);
         clear('C')
         fclose(file);
         disp(contador)
         contador=contador+1;
      end
      clear('A')
   end
end
disp('Construccion de archivos arff: Terminada');
```

```
function fprintmat(C, file, idio, ii, jj, tam1)
%global C file idio ii jj tam1
newLanguage_CellString= evalin ('base','idiomas_CellString');
languages_CellString= evalin ('base','languages_CellString');
t=size(C);
for i=1:t(1)
   pos=char(newLanguage_CellString{1});
   if i>tam1
      pos=char(languages_CellString{jj});
   end
   for j=1:t(2)
      fprintf(file, '%.9f',C(i,j));
      if j==t(2)
         fprintf(file, ', %s\n', pos);
      else
         fprintf(file, ', ');
      end
   end
end
```

4. Código para caracterizar una nueva instancia.

```
function newInstance( sessionId , fileName)
newFilePath=[char(sessionId) '/' char(fileName)];
NameFileAtri=[char(sessionId) '/' char(fileName(1:end-4)) '.arff'];
seconds_number=50;
porcentaje=0.08;
descomposition_level=1;
[signal,sampling_rate]=wavread(newFilePath);
for k=40:seconds_number-1
   segemented_signal=signal(k*sampling_rate+1:(k+1)*sampling_rate);
   [C,L]=wavedec(segemented_signal,descomposition_level,'db2');
   coef_number=L(1);
   [sorted_wavelets, actual_indixes] = sort(abs(C(1:coef_number)),'descend'); % se
   truncated_wavelets=zeros(coef_number,1);
   for q=1:coef_number*porcentaje % size(orderedRandomSignal)/fraction
      truncated_wavelets(actual_indixes(q))=C(actual_indixes(q));
   end
   temp=char( char(fileName));
   temp=temp(1:end-4);
   directorio_wavelets=[char(sessionId) '/' temp  int2str(k+1) '.wavelet'];
   archivo_escritura=fopen(directorio_wavelets,'w');
   fprintf(archivo_escritura,'%.9f \n',truncated_wavelets);
   fclose(archivo_escritura);
end
```

```
m=[];
for k=41:seconds_number
   rn=[char(sessionId) '/' char(fileName(1:end-4)) int2str(k) '.wavelet'];
   archivo=fopen(rn,'r');
   c=fscanf(archivo,'%f');
   m=[m c];
end
m=m';
media=mean(m);
desviacion=std(m);
maximo=max(m);
minimo=min(m);
curtosis=kurtosis(m);
obliquidade=skewness(m);
atri=[media desviacion maximo minimo curtosis obliquidade];
fil=fopen(NameFileAtri,'w+');
atributos=char('Media','Desviacion','Maximo','Minimo','Curtosis','Obliquidade');
a=cellstr(atributos);
primero='@RELATION lenguajes';
fprintf(fil,'%s\n',primero);
for i=1:6     for j=1:4001
       linea=['@ATTRIBUTE' ' ' a{i,:} int2str(j) ' ' 'REAL'];
       fprintf(fil,'%s\n',linea);
   end
   fprintf(fil,'\n');
end
linea='@ATTRIBUTE class  { ? }';
fprintf(fil,'%s\n',linea);
linea='@DATA';
fprintf(fil,'%s\n\n',linea);
for j=1:length(atri)
   fprintf(fil,'%.9f, ',atri(j));
end
fprintf(fil,' ?');
fprintf(fil,'\n');
clear('media');clear('desviacion');clear('maximo');clear('minimo');
clear('curtosis');
clear('obliquidade');
clear('m');
m=[];
fclose(fil);
clear all;
```

ANEXO B

CÓDIGO JAVA PARA LA CAPTURA DE MUESTRAS DE AUDIO.

1. Código de la clase para generar la interfaz gráfica de la aplicación que genera los modelos a partir de las muestras de cada idioma.

```
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.*;

import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.filechooser.FileFilter;



class PrincipalWindow extends JFrame {
 /**
          *
          */
         private static final long serialVersionUID = 1L;
         private JList list ;
         private DefaultListModel listModel;
         private JButton aceptarBtn = new JButton("Crear Modelo");
         private JLabel languageLbl = new JLabel("Language");
         private JLabel numberOFRows= new JLabel("0");
         private JTextField languageTxt = new JTextField(10);
         private JButton exploreBtn = new JButton("Agregar");
         private JButton deleteBtn = new JButton("Quitar");
         private LanguageAgregator languageAgregator= new LanguageAgregator();

         public PrincipalWindow() {
                  this.setTitle("My Empty Frame");
                  this.setSize(600,400); // default size is 0,0
                  this.setLocation(10,200); // default is 0,0 (top left corner)
                  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
            //list = new JList(new Object [3]); //data has type Object[]
            listModel = new DefaultListModel();


          //Create the list and put it in a scroll pane.
            list = new JList(listModel);
list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
  list.setSelectedIndex(0);
  list.setLayoutOrientation(JList.VERTICAL);
  //list.addListSelectionListener(this);
  list.setVisibleRowCount(5);
  JScrollPane listScrollPane = new JScrollPane(list);
 exploreBtn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub
    //Create a file chooser
    final JFileChooser fc = new JFileChooser();
    fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
    fc.setAcceptAllFileFilterUsed(false);
    fc.addChoosableFileFilter(new WavFileFilter());
    fc.setMultiSelectionEnabled(true);
    //In response to a button click:
    int returnVal = fc.showOpenDialog(null);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
            File [] selectedFileS = fc.getSelectedFiles();
            System.out.println(selectedFileS[0].getName());
            for (File f: selectedFileS){
                    listModel.addElement(f);
            }
            numberOFRows.setText(Integer.toString(listModel.getSize()));
                            }
                    }
            } );
            aceptarBtn.addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent arg0) {
                            // TODO Auto-generated method stub
                            String languageName=languageTxt.getText().trim();
                            if(languageName.length()==0){
            JOptionPane.showMessageDialog(getContentPane(),"Intruce el nombre del
            Lenguaje.");
            return;
                            }
                            languageAgregator.validateFiles(list, listModel);

                            if(languageAgregator.hasErrorFiles()){
```

```
        JOptionPane.showMessageDialog(getContentPane(),languageAgregator.getErrorMe
ssage());
                                        return;
                                }


        languageAgregator.agregateLanguageWavFiles(languageName,listModel);
                }
                } );
                list.addListSelectionListener(new ListSelectionListener() {
                        @Override
                        public void valueChanged(ListSelectionEvent e) {
                                // TODO Auto-generated method stub
                                deleteBtn.setEnabled(!list.isSelectionEmpty());
        numberOFRows.setText(Integer.toString(listModel.getSize()));
                        }
                });
                deleteBtn.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                // TODO Auto-generated method stub
                                int selectedIndexes [] = list.getSelectedIndices();
                                for(int index=selectedIndexes.length-1;index>=0;index--){
                                        listModel.remove(selectedIndexes[index]);
                                        //list.remove(selectedIndexes[index]);
                                }
                        }
                });
        list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION)
;
                list.setLayoutOrientation(JList.HORIZONTAL_WRAP);
                list.setVisibleRowCount(-1);

                JScrollPane listScroller = new JScrollPane(list);
                listScroller.setPreferredSize(new Dimension(250, 80));
                JPanel northPanel = new JPanel();
                northPanel.add(languageLbl);
                northPanel.add(languageTxt);
                northPanel.add(aceptarBtn);
                northPanel.add(exploreBtn);
                deleteBtn.setEnabled(false);
                northPanel.add(deleteBtn);

                this.add(northPanel,BorderLayout.NORTH);
                this.add(listScroller,BorderLayout.CENTER);
                this.add(numberOFRows,BorderLayout.SOUTH);
```

```
        this.setVisible(true);
  }

  public static void main(String[] args) {
    JFrame f = new PrincipalWindow();

  }
}

class WavFileFilter extends FileFilter
{



        @Override
        public String getDescription() {
                // TODO Auto-generated method stub
                return "Wav Files Only";
        }

        @Override
        public boolean accept(File pathname) {
                // TODO Auto-generated method stub
                if (pathname.isDirectory()) {
            return true;
           }

           String extension = Utils.getExtension(pathname);
           if (extension != null) {
              if (extension.equals(Utils.WAV_EXTENTION)) {
                    return true;
              } else {
                 return false;
              }
           }

           return false;
        }

}
```

2. Clase encargada de copiar los archivos al directorio de trabajo de Matlab y llamar las rutinas mostradas en el anexo A.

```
import java.io.BufferedWriter;
import java.io.File;
```

```
import java.io.FileWriter;
import java.io.IOException;

import javax.swing.DefaultListModel;
import javax.swing.JList;

import matlabcontrol.MatlabProxy;
import matlabcontrol.MatlabProxyFactory;
import matlabcontrol.MatlabProxyFactoryOptions;

public class LanguageAgregator {

	private boolean errorFiles;
	private String errorMessage;

	public void agregateLanguageWavFiles(String languageName,DefaultListModel
listModel){
		try
		{


			String wavFilesLanguageDirectory =
Utils.WAV_FILES_DIRECTORY+languageName+"/";
			new File(wavFilesLanguageDirectory).mkdir();
			String waveletsFilesLanguageDirectory =
Utils.WAVELET_FILES_DIRECTORY+languageName+"/";
			new File(waveletsFilesLanguageDirectory).mkdir();

			FileWriter fstream = new
FileWriter(Utils.PRIMER_PASO+languageName+".txt",true);
			BufferedWriter out = new BufferedWriter(fstream);
			int filesNumber =listModel.getSize();
			for(int i=0;i<filesNumber;i++){
				File file=((File)listModel.getElementAt(i));
				FileCopy.copy(file.getAbsolutePath(),
wavFilesLanguageDirectory+file.getName());
				out.write(file.getName()+"\n");
			}
			out.close();


			// Create file
			fstream = new FileWriter(Utils.NEW_LANGUAGES_LIST,false);
			out = new BufferedWriter(fstream);
			out.write(languageName+"\n");
			//Close the output stream
			out.close();
```

```
                    MatlabProxyFactoryOptions options = new
MatlabPrxoxyFactoryOptions.Builder()
                .setUsePreviouslyControlledSession(false).setHidden(true)
                .setMatlabLocation(null).build();
                    MatlabProxyFactory factory = new MatlabProxyFactory(options);
                    MatlabProxy proxy = factory.getProxy();

                    proxy.eval("cd '"+Utils.PRIMER_PASO+"'");
                    proxy.eval("principal");
                    proxy.eval("cd ..");
                    proxy.eval("cd '"+Utils.SEGUNDO_PASO+"'");
                    proxy.eval("atributos");


                    proxy.eval("cd ..");
                    proxy.eval("cd '"+Utils.TERCER_PASO+"'");
                    proxy.eval("combinar");


                    FileWriter fstream2 = new
FileWriter(Utils.LANGUAGES_LIST,true);
                    BufferedWriter out2 = new BufferedWriter(fstream2);
                    out2.write(languageName+"\n");
                    //Close the output stream
                    out2.close();
                    proxy.eval("quit");

            }catch (Exception e){//Catch exception if any
                    System.err.println("Error: " + e.getMessage());
            }

    }

    public void validateFiles(JList list, DefaultListModel listModel) {
            this.errorFiles=false;
            int filesNumber = listModel.getSize();
            if (filesNumber != 50) {
                    setErrorFiles(true);
                    errorMessage = new String("Se necesitan al menos 50 archivos");
                    return;
            }

            for (int i = 0; i < filesNumber; i++) {
                    System.out.println(((File) listModel.getElementAt(i))
                                    .getAbsolutePath());
                    WavFile wavFile;
```

```
                        try {
                                File file = ((File) listModel.getElementAt(i));
                                wavFile = WavFile.openWavFile(file);
                                if (wavFile.getNumFrames() <
Utils.WAV_MINIMUM_SIZE) {
                                        list.setSelectedIndex(i);
                                        setErrorFiles(true);
                                        errorMessage = new String("Los archivos
seleccionados no cuentan con la longitud necesaria.");
                                        return;
                                }
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (WavFileException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                }
        }

        public String getErrorMessage() {
                return errorMessage;
        }

        public void setErrorMessage(String errorMessage) {
                this.errorMessage = errorMessage;
        }

        public boolean hasErrorFiles() {
                return errorFiles;
        }

        public void setErrorFiles(boolean errorFiles) {
                this.errorFiles = errorFiles;
        }

}
```

3. Código de la clase que valida el tipo de archivo, el tamaño mínimo y la cantidad mínima de muestras a procesar.

```
import java.io.File;
import java.io.IOException;
```

```
import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.JOptionPane;


public class FileValidator {
	private boolean success=true;
	private String errorMessage;


	public void validate(JList list,DefaultListModel listModel){
		int filesNumber =listModel.getSize();
		if (filesNumber!=50){
			success=false;
			errorMessage= new String("Se necesitan al menos 50 archivos");
			return;
		}

		for(int i=0;i<filesNumber;i++)
		{

	System.out.println(((File)listModel.getElementAt(i)).getAbsolutePath());
			WavFile wavFile;
			try {
				File file=((File)listModel.getElementAt(i));
				wavFile = WavFile.openWavFile(file);

	if(wavFile.getNumFrames()<Utils.WAV_MINIMUM_SIZE){
					list.setSelectedIndex(i);
					success=false;
					errorMessage= new String("Los archivos
seleccionados no cuentan con la longitud necesaria.");
					return;
				}
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (WavFileException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		}
	}


	public String getErrorMessage() {
```

```
		return errorMessage;
	}
	public void setErrorMessage(String errorMessage) {
		this.errorMessage = errorMessage;
	}
	public boolean isSuccess() {
		return success;
	}
	public void setSuccess(boolean success) {
		this.success = success;
	}


}
```

4. Servlet que captura la nueva instancia desde un explorador web.

```
// Import required java libraries
import java.io.*;
import java.util.*;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import matlabcontrol.MatlabProxy;
import matlabcontrol.MatlabProxyFactory;
import matlabcontrol.MatlabProxyFactoryOptions;

import org.apache.tomcat.util.http.fileupload.FileItem;
import org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory;
import org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload;



public class UploadServlet extends HttpServlet {

  private boolean isMultipart;
  private String filePath;
  private int maxFileSize = 5000 * 1024;
  private int maxMemSize = 4000 * 1024;
  private File file ;

  public void init( ){
```

```
  // Get the file location where it would be stored.
  filePath =
        getServletContext().getInitParameter("file-upload");
}
public void doPost(HttpServletRequest request,
          HttpServletResponse response)
         throws ServletException, java.io.IOException {
  // Check that we have a file upload request
  isMultipart = ServletFileUpload.isMultipartContent(request);
  response.setContentType("text/html");
  java.io.PrintWriter out = response.getWriter( );
  if( !isMultipart ){
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet upload</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<p>No file uploaded</p>");
    out.println("</body>");
    out.println("</html>");
    return;
  }
  DiskFileItemFactory factory = new DiskFileItemFactory();
  // maximum size that will be stored in memory
  factory.setSizeThreshold(maxMemSize);
  // Location to save data that is larger than maxMemSize.
  factory.setRepository(new File("temp"));

  // Create a new file upload handler
  ServletFileUpload upload = new ServletFileUpload(factory);
  // maximum file size to be uploaded.
  upload.setSizeMax( maxFileSize );

  try{
  // Parse the request to get file items.
  List fileItems = upload.parseRequest(request);

  // Process the uploaded file items
  Iterator i = fileItems.iterator();

  out.println("<html>");
  out.println("<head>");
  out.println("<title>Servlet upload</title>");
  out.println("</head>");
  out.println("<body>");
  while ( i.hasNext () )
  {
```

```
        FileItem fi = (FileItem)i.next();
        if ( !fi.isFormField () )
        {
          // Get the uploaded file parameters
          String fieldName = fi.getFieldName();
          String fileName = fi.getName();
          String contentType = fi.getContentType();
          boolean isInMemory = fi.isInMemory();
          long sizeInBytes = fi.getSize();
          // Write the file
          new File(filePath + "/"+request.getRequestedSessionId() +"/").mkdir();
          if( fileName.lastIndexOf("\\") >= 0 ){
            file = new File( filePath + "/"+request.getRequestedSessionId() +"/"+
            fileName.substring( fileName.lastIndexOf("\\"))) ;
          }else{
            file = new File(  filePath + "/"+request.getRequestedSessionId() +"/"+
            fileName.substring(fileName.lastIndexOf("\\")+1)) ;
          }
          fi.write( file ) ;
          out.println("Uploaded Filename: " + fileName + "<br>");
        }
      }
      MatlabProxyFactoryOptions options = new MatlabProxyFactoryOptions.Builder()
              .setUsePreviouslyControlledSession(false)
              .setHidden(true)
              .setMatlabLocation(null).build();
                    MatlabProxyFactory factoryMatlab = new MatlabProxyFactory(options);
             MatlabProxy proxy = factoryMatlab.getProxy();

                    proxy.eval("cd '"+filePath+"'");
                    proxy.feval("newInstance",request.getRequestedSessionId(),file.getName());

                    LanguageIdentification li = new LanguageIdentification();
                    li.startLID(request.getRequestedSessionId(),file.getName());
                    Map <String,Integer>orderedParticipantScores = new
HashMap<String,Integer>();

                    orderedParticipantScores.putAll(li.getParticipantScores());
                    out.print("<ul>");
                    for (String participant : orderedParticipantScores.keySet()){
                          System.out.println(participant+" = "+
orderedParticipantScores.get(participant));
                          out.print("<li>"+participant+" = "+
orderedParticipantScores.get(participant)+"</li>");
                    }out.print("</ul>");

      out.println("</body>");
```

```
    out.println("</html>");
    proxy.eval("quit");
  }catch(Exception ex) {
     System.out.println(ex);
  }
  }
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, java.io.IOException {

      throw new ServletException("GET method used with " +
            getClass( ).getName( )+": POST method required.");
  }
}
```

5. Clase que realiza la Identificación del idioma de la nueva instancia.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;


public class LanguageIdentification {

        private ArrayList<String> languageNames = new ArrayList<String>();
        private Map <String,Integer>participantScores = new HashMap<String,Integer>();
        public LanguageIdentification(){
                this.languageNames=this.getLanguageNames();
        }


        public void startLID(String sessionId, String fileName){
                RoundRobinTournament rrt=new
RoundRobinTournament(this.languageNames,sessionId,fileName);
                participantScores=rrt.doTournament(sessionId, fileName);
        }

        public static void main(String []args){
                LanguageIdentification lid= new LanguageIdentification();
                lid.startLID("F1A6490D04BE34EF99FF14BD134E61F7", "FA-
8.story.wav");
```

```
	}

	public void setLanguageParticipants(ArrayList<String> languageNames) {
		this.languageNames = languageNames;
	}

	public ArrayList<String> getLanguageNames(){

		String listNamesPath=Utils.LANGUAGES_LIST;

		FileReader fileReader;
		try {
			fileReader = new FileReader(new File (listNamesPath));
			BufferedReader bufferedReader = new BufferedReader(fileReader);
			String line = bufferedReader.readLine();
			while(line!=null && line.length()!=0){
				//System.out.println(line);
				this.languageNames.add(line);
				line = bufferedReader.readLine();
			}
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}


		return this.languageNames;
	}


	public Map <String,Integer> getParticipantScores() {
		return participantScores;
	}


	public void setParticipantScores(Map <String,Integer> participantScores) {
		this.participantScores = participantScores;
	}
}
```

6. Clase que realiza el torneo Round Robin entre los idiomas para determinar un ganador.

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import javax.swing.Box.Filler;

import weka.attributeSelection.InfoGainAttributeEval;
import weka.attributeSelection.Ranker;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;


public class RoundRobinTournament {
	private Map <String,Integer>participantScores = new HashMap<String,Integer>();
	private ArrayList<String> participantNames = new ArrayList<String>();
	private String newInstanceName= new String();
	private String sessionId= new String();

	public ArrayList<String> getParticipantNames() {
		return participantNames;
	}


	public void setParticipantNames(ArrayList<String> participantNames) {
		this.participantNames = participantNames;
	}


	public RoundRobinTournament(ArrayList<String> participantNames,String
sessionId,String newInstanceName){
		this.participantNames=participantNames;
		this.participantScores=InitializeScores();
	}


	public Map<String,Integer> InitializeScores(){
		Map <String,Integer> pp = new HashMap<String,Integer>();
```

```
		for(String participantName : this.getParticipantNames()){
			pp.put(participantName, new Integer(0));
		}
		return pp;
	}

	public Map<String, Integer> getParticipantsScores() {
		return participantScores;

	}
	public void setParticipantsScores(Map <String,Integer>participantsScores) {
		this.participantScores = participantsScores;
	}
	public Map<String, Integer> doTournament(String sessionId, String fileName) {
		String
newInstancePath=Utils.MATLAB_DIRECTORY+"/"+sessionId+"/"+fileName.substring(0
,fileName.length()-3)+"arff";
		ArrayList<String> participantNames = this.getParticipantNames();
		int participantNumber= participantNames.size();
		for (int i=participantNumber-1;i>=0;i--){
			for(int j=i-1;j>=0;j--){
				System.out.println(participantNames.get(i)+"-
"+participantNames.get(j)+".arff");
				String winner = singleMatch(participantNames.get(i),
participantNames.get(j),newInstancePath);
				Integer currentWinnerScore
=this.getParticipantsScores().get(winner);
				currentWinnerScore++;
				this.getParticipantsScores().put(winner, currentWinnerScore);
				System.out.println("Ganador: "+winner);
			}
		}
		return getParticipantsScores();
		// TODO Auto-generated method stub

	}
	public String singleMatch(String firstGamer,String secondGamer, String
newInstancePath){
		String wekaFilesPath = Utils.WEKA_FILES+firstGamer+"-
"+secondGamer+".arff";
		try {
			String winner ;
			int cls=clasificar(wekaFilesPath,newInstancePath);

			if(cls==1)
				winner= secondGamer;
			else winner=firstGamer;
```

```
                return winner;
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return null;
        }
        public int clasificarRon(String nombre_archivo_entrenamiento, String
newInstancePath) throws Exception{


            System.out.println("Clasifcacion sin ganancia de informacion");
     //esta funcion se encarga de cargar el archivo arff, filtrarlo y clasificarlo
     Classifier C = seleccionarClassificador(1);//Se elige el clasificador
      Instances entrenamiento = cargarInstancias(nombre_archivo_entrenamiento);//Se
cargan las instancias
      entrenamiento.setClassIndex(entrenamiento.numAttributes() - 1);
      C.buildClassifier(entrenamiento);
     Instances unlabeled= new Instances(new BufferedReader(new
FileReader(newInstancePath)));

     unlabeled.setClassIndex(unlabeled.numAttributes() - 1);



     double cls = C.classifyInstance(unlabeled.instance(0));
    // System.out.println(cls);
     //System.out.println(entrenamiento.classAttribute().value((int)cls));
     return (int)cls;
   }
        public int clasificar(String nombre_archivo_entrenamiento, String
newInstancePath) throws Exception{


            System.out.println("Clasifcacion con ganancia de informacion");
          //esta funcion se encarga de cargar el archivo arff, filtrarlo y clasificarlo
          Classifier C = seleccionarClassificador(1);//Se elige el clasificador
           Instances entrenamiento =
cargarInstancias(nombre_archivo_entrenamiento);//Se cargan las instancias
           Instances filtradas;//Donde se depositaran una vez filtradas las instancias
           boolean terminado, cargado;//Para verificacion
           InfoGainAttributeEval filtro =new InfoGainAttributeEval();//Filtro de
seleccion de atributos
           Ranker busqueda = new Ranker();//Ranker, enlistador, buscador
           //Se establecen los parametros del ranker
           busqueda.setGenerateRanking(true);//por dfecto
           busqueda.setNumToSelect(-1);//por defecto
```

```
                busqueda.setStartSet("");//por defecto
                busqueda.setThreshold(0.0);//se elige un umbral de cero aportacion
                //Objeto de seleccion de atributos
                weka.filters.supervised.attribute.AttributeSelection Seleccion= new
weka.filters.supervised.attribute.AttributeSelection();
                //Se establecen sus parametros
                Seleccion.setEvaluator(filtro);//Evaluador, tipo de filtro ganacia de
informacion
                Seleccion.setSearch(busqueda);//Buscada, mediante ranker
                cargado=Seleccion.setInputFormat(entrenamiento);//se copia el formato de las
instancias, devuelve true si fue exitoso
                //se copian las instancias, ya establecido el formato
                for(int i=0;i<entrenamiento.numInstances();i++)
                {
                   Seleccion.input(entrenamiento.instance(i));

                }

                //Se hace el filtrado
                terminado=Seleccion.batchFinished();
                filtradas=Seleccion.getOutputFormat();//se copia el formato a filtradas
                //se copian las instancias a filtradas
                while(Seleccion.numPendingOutput()>0)
                {
                   filtradas.add(Seleccion.output());
                }
               //Se colocan aleatoriamente
               //filtradas.randomize(new Random(1));
               C.buildClassifier(filtradas);
               Instances newInstance= new Instances(new BufferedReader(new
FileReader(newInstancePath)));
               /*Instances truncateNewInstance = new Instances();
               truncateNewInstance.
               */
               Instance inst = new Instance(filtradas.numAttributes());
               Instances instances;
               newInstance.setClassIndex(newInstance.numAttributes() - 1);
               System.out.println(filtradas.numAttributes());
               int attrIndixes [] = new int [entrenamiento.numAttributes()];
               for(int i=0;i<filtradas.numAttributes();i++){
                     String attributeName=filtradas.attribute(i).name();
                     for (int j = 0; j < Utils.STATISTICAL_MEASURES.length; j++) {
                                  if
(attributeName.contains(Utils.STATISTICAL_MEASURES[j])) {
                                        String subfixNumber =
attributeName.substring(Utils.STATISTICAL_MEASURES[j].length());
int index=(Integer.valueOf(subfixNumber)+(j*4001))-1;
```

```
                //System.out.println("Name: "+filtradas.attribute(i).name());
                //System.out.println("Valores: "+newInstance.firstInstance().value(index));
                //System.out.println("Nombre Attributo Entrenamiento
"+filtradas.attribute(i).name());
                //System.out.println("Nombre Attributo Entrenamiento
"+newInstance.attribute(index).name());
                inst.setValue(filtradas.attribute(i), newInstance.firstInstance().value(index));

        //truncateNewInstance.insertAttributeAt(newInstance.attribute(i), i);
                                }
                        }

            }
            inst.setDataset(filtradas);
            inst.enumerateAttributes();

            /*for (int i = 0; i < attrIndixes.length; i++) {
                  if(attrIndixes[i]==0)
                        unlabeled.deleteAttributeAt(attrIndixes[i]);
               }*/

            double cls =C.classifyInstance(inst);
            //C.
            System.out.println();
            return (int)cls;
          }

                public static Classifier seleccionarClassificador(int i) {
                    //Eleccion del clasificador
                    switch(i) {
                       //case 0: return new weka.classifiers.trees.j48.J48();
                       case 1: return new weka.classifiers.bayes.NaiveBayes();
                       case 2: return new weka.classifiers.bayes.BayesNet();
                       //default: return new weka.classifiers.bayes.BayesNetK2();
                    }
                                return null;
                  }

public static Instances cargarInstancias(String nombre_archivo) throws Exception{
                    Instances I;
                    //se carga el archivo de instancias
                    I = new Instances(new FileReader(nombre_archivo));
                    I.setClassIndex(I.numAttributes()  - 1);

                    return I;
                  }
```

```
public static Evaluation ValidacionCruzada(Classifier C, Instances entrenamiento, int
numfolds)throws Exception {
                    //objeto de evaluation, inicializado con las instancias de entrenamiento
filtradas
                     Evaluation E = new Evaluation(entrenamiento);
                     //Ejecucion de la clasificacion mediante validacion cruzada usando
Naive Bayes
                     //E.crossValidateModel(C, entrenamiento, numfolds);
                     return E;
                  }
}
```