INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Un Método para la Multiclasificación de Idiomas Usando la Transformada Wavelet

PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS EN CIENCIAS DE LA COMPUTACION

PRESENTA **I.S.C. CÉSAR MEDINA TREJO**

DIRECTOR **DR. ARTURO HERNÁNDEZ RAMÍREZ**

Cd. Madero, Tamps., México Abril 2011

"2011, Año del Turismo en México"

SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

Cd. Madero, Tamps; a 22 de Marzo de 2011

OFICIO No.: U5. 107/11
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. CÉSAR MEDINA TREJO
P R E S E N T E

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 MÉTODO PARA LA MULTICLASIFICACIÓN DE IDIOMAS USANDO LA TRANSFORMADA WAVELET"

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.

A T E N T A M E N T E
"Por mi Patria y por mi Bien"

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

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

c.p.: Archivo

MYCHC 'NMCO 'avbc*

Ave. 10. De Mayo y Sor Juana I. De la Cruz, Col. Los Mangos, C.P. 89440 Cd. Madero, Tam.
Tels. (833) 3 57 48 20. Fax: (833) 3 57 48 20, Ext. 1002, email: itcm@itcm.edu.mx
www.itcm.edu.mx

DECLARACIÓN DE ORIGINALIDAD

Declaro que este documento de tesis es un trabajo original, donde en las citas incluidas y referencias, se indica explícitamente los datos de las publicaciones asi como sus autores, incluidos en las referencias bibligráficas.

Acepto la total responsabilidad en caso de infrigir con las leyes de derechos de terceros, de cualquier reclamación relacionada con los derechos de propiedad intelectual, que se puedan derivar en este documento de tesis, exonerando de la responsabilidad a mi director de tesis, así como al Instituto Tecnológico de Ciudad Madero.

13 de Abril de 2011, Cd. Madero, Tamps.

_____

Ing. César Medina Trejo

DEDICATORIA:

Dedico este trabajo a mi padre César Medina de Leon, a mi madre Maria Cristina Estela Trejo Salas, asi como a mi hermano Jesús Medina Trejo.

AGRADECIMIENTOS:

Agradezco primeramente a mis padres, sin ellos no tendría el motor para seguir adelante, a mi madre, por sus noches de desvelo y preocupación al mismo tiempo que yo me desvelaba, por todos sus consejos, su confianza, su amor, por depositar su fe en mí. A mi padre, por todos los consejos que me ha dado, que siempre los llevo conmigo, por su confianza en mi, agradezco todos los esfuerzos para que no me haya faltado nada durante lo que llevo de mi vida estudiantil, agradezco el ejemplo que me da como persona, un modelo a seguir. Por eso y muchas cosas más, gracias a mis padres.

Agradezco a mi hermano, porque se que incondicionalmente me brinda todo su apoyo asi como todos los consejos, compañía que me ha brindado,su cariño y por todos los días que estoy seguro se ha preocupado por mi.

Agradezco a mi director de tesis, el Dr. Arturo Hernández Ramírez, por ser un gran maestro, que me ha enseñado muchas cosas, por todos sus consejos de los cuales he aprendido mucho, gracias por haber depositado su confianza en mi para poder sacar adelante esta tesis, sin el esta tesis no tendría forma, lo considero una gran persona y maestro, gracias por ser una persona accesible, que me atendió en cualquier duda que tenia, por muchas cosas mas, gracias.

Agradezco a todos mis compañeros de generación, con los cuales convivi estos años, que tal vez puede tener roces con algunos, con otros no, quiero agradecerles a todos por igual la compañía brindada en este tiempo, y siempre los tendre presente, no los olvidare, son las personas que le dieron ese toque especial a pasar el tiempo en el instituto.

Agradezco a mis grandes amigos fuera del instituto, por su comprensión y apoyo, cuando no he asistido a algunas reuniones, por estar ocupado trabajando, ellos son mis puntos de apoyo cuando necesito hablar con alguien o distraerme un poco: Ruben, Diego F,. Diego M, Victor, Dorian, a todos los que pudiera olvidar en este momento, gracias.

Agradezco a mi novia Mercedes, porque gracias a ella mejore como persona, por todos sus consejos y apoyo incondicional brindado, sin ella me hubiera quebrantado en algunos momentos de crisis emocional, sin su apoyo no hubiera podido seguir delante de la forma tan positiva en que lo he hecho, muchas gracias.

Resumen:

El problema de la identificación automatica de lenguas habladas (LID) es identificar el idioma en que un hablante se esta comunicando, a partir de una muestra de voz. En la actualidad se ha trabajado poco con la Transformada Wavelet para abordar este problema, pero los resultados que se muestran indican un porcentaje de clasificación alto.

Los trabajos que usan la Transformada Wavelet para resolver LID, se basan en una metodología de biclasificación, no se ha trabajado en una metodología de multiclasificación con la Transformada Wavelet.

El presente trabajo agrega dos medidas estadísticas (curtosis y skewness) a los biclasificadores, mejorando el porcentaje de clasificacion promedio de 93% a 96%, mostrando que esta nueva caracterización es efectiva. Se propone una metodología para la multiclasificacion de idiomas basado en tecnicas de torneo, a partir de un conjunto de biclasificadores. Los resultados obtenidos en la multiclasificación indican que es una buena técnica de clasificación, alcanzando en algunos casos un porcentaje de clasificación del 100% para algunos idiomas.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Capítulo 1 Introducción

1.1 Antecedentes

El problema principal en resolver la tarea de la identificación de lenguas habladas (LID) es reducir la complejidad del lenguaje humano de tal forma que un algoritmo pueda identificarlo a partir de una muestra de audio corta [Navrátil]. Tiene un rango variado de aplicaciones, como pueden ser: a) preprocesamiento a sistemas de traducción multilingüe, para empezar a traducir en el idioma necesario, o para la comunicación de los hablantes con el personal adecuado que pueda entender su idioma (llamadas de emergencia), b) también en uso de interfaces de voz que eligen el idioma de habla del usuario, c) la interacción multilingüe, donde se necesita un traductor que trabaje en tiempo real para que dos hablantes se comuniquen en sus respectivos idiomas.

En la actualidad existen diferentes acercamientos para resolver sistemas LID [Schultz y Kirchoff 2006], los más importantes son:

- Acústico
- Fonotáctico
- Prosódico
- LVCSR (Grandes identificadores de habla de vocabulario continuo)

Los acercamientos más precisos son el fonotáctico y LVCSR, dado que utilizan información fonotáctica, información detallada de cada idioma, pero implica tener un alto costo de cómputo y recursos lingüísticos altos, para obtener características más finas de las lenguas.

Para las lenguas que no cuentan con una base lingüística solida (por ejemplo, lenguas indígenas), se usan acercamientos como el acústico, el cual está basado en el ritmo de los lenguajes hablados, este trabajo utiliza dicho enfoque, se aplica la Transformada

Wavelet como técnica de procesamiento propuesta, y enfatiza el uso de muestras cortas de habla como su principal ventaja.

Vargas-Martínez sigue la misma línea de trabajo de Reyes-Herrera quien fue la primera en utilizar la transformada wavelet en esta línea de investigación, Vargas, a diferencia de Reyes, hace uso de medidas estadísticas y una segmentación de la señal de habla dividiéndola en segundos, con el fin de obtener un menor número de coeficientes y mejorar el costo computacional.

Este trabajo es una continuación de lo propuesto por Vargas, se añaden nuevas medidas estadísticas como la curtosis y skewness, con el propósito de mejorar los resultados, pero la principal aportación es el uso de la técnica de multiclasificacion, usando métodos de torneo, un enfoque poco explorado en esta línea de investigación.

1.2 Definición del problema

El reconocimiento automático del lenguaje hablado por medio de máquinas ha sido la meta de investigación por varias décadas. Durante el paso del tiempo, se han realizado diferentes estudios en este campo usando una amplia gama de acercamientos, obteniendo resultados variados.

Se detectaron otros problemas que están ligados al reconocimiento del habla, los cuales son: Reconocimiento de emociones, Identificación del hablante, la detección de voz activa y la identificación de lenguas habladas, siendo esta última donde se enfocará el proyecto de tesis.

El problema de la identificación automática de lenguas, es identificar el lenguaje hablado por un hablante desconocido mediante un pequeño extracto del habla [Muthusamy 1992]. En los últimos años se ha elevado la importancia de este problema en particular en el ámbito del reconocimiento del habla, debido a la globalización, en el uso de interfaces multilingües en distintas aplicaciones como en aeropuertos, llamadas de emergencia.

1.3 Justificación

En la actualidad no existen muchos sistemas que identifiquen automáticamente las lenguas por medio de clasificación por torneos, la mayoría de los sistemas usan métodos de biclasificación. Se busca mejorar la metodología de Vargas Martínez, la cual usa biclasificadores, debido a que la multiclasificación implementada se basa en un conjunto de estos, y al mejorarlos, por consecuencia la multiclasificación será más precisa. Dado que se trabaja con una caracterización puramente acústica, en un futuro pretende aplicarse a lenguas indígenas de México.

1.4 Objetivos

Objetivo general

Realizar una multiclasificación de los diferentes idiomas encontrados en la base de datos estándar OGI_TS mediante métodos de torneo, usando la técnica de la Transformada Wavelet Daubechies.

Objetivos específicos

- Entender cómo funciona la transformada wavelet.
- Aplicar un diferente tipo de transformada wavelet a la usada previamente en el trabajo base.
- Emplear nuevas medidas estadísticas para caracterizar la entrada.
- Implementar los procesos de la transformada wavelet en los programas Matlab y Praat.
- Hacer biclasificación usando otra transformada wavelet como la Dbn para n>2.
- Implementar una multiclasificación usando técnicas de clasificación por torneos.

1.5 Alcances y limitaciones

Limitaciones

- Implementar módulos de un sistema de reconocimiento automático de lenguas
- Se usaran 9 lenguas de la base de datos OGI-TS, las cuales son: Inglés, Alemán, Español, Japonés, Tamil, Mandarín, Coreano, Farsi y Vietnamita.
- Uso de características acústicas.

Alcances

- Construcción de una metodología de identificación de lenguas.
- Experimentación con diferentes tipos de wavelets.
- Multiclasificación.

1.6 Organización de la Tesis

La tesis está organizada de la siguiente manera:

En el capítulo 2: Marco teórico, se muestran los fundamentos de la transformada wavelet, así como conceptos de minería de datos y las nuevas medidas estadísticas empleadas.

En el capítulo 3: Estado del arte, se hace una revisión sobre los principales acercamientos que existen para resolver LID, las personas que han trabajado en ellos, al igual que los trabajos relacionados directamente que usan la transformada wavelet.

En el capítulo 4: Metodología, aquí se mostrará el procedimiento para obtener los biclasificadores y la forma en que estos son usados para simular los torneos en la multiclasificación.

En el capítulo 5: Experimentación y Resultados: En este capítulo se mostraran todas las pruebas realizadas y el promedio de clasificación obtenido de cada una.

En el capítulo 6: Conclusiones y Trabajos Futuros: Se mencionarán las conclusiones pertinentes al presente trabajo así como las recomendaciones para mejorarlo.

Anexo A: En este anexo se mostraran las tablas referentes a la justificación de NaiveBayes y el uso de diferentes transformadas wavelet, para justificar las tablas de promedios obtenidas en el capítulo 5.

Anexo B: Se mostrarán todos los códigos usados en el proceso de biclasificación, así como la multiclasificación.

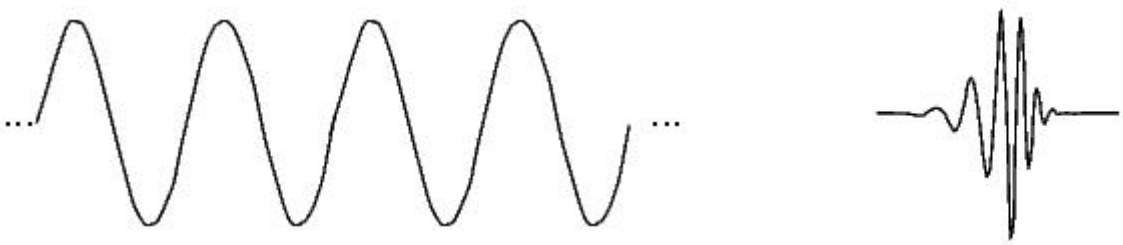
Capítulo 2 Marco Teórico

Este capítulo trata los conceptos referentes a la Transformada Wavelet, el fundamento necesario de minería de datos para realizar el proceso de caracterización, además de los conceptos referentes a las nuevas medidas estadísticas usadas en la metodología. Los conceptos matemáticos de la Transformada de Fourier, así como una perspectiva histórica de porque se usan la transformada wavelet sobre la transformada de Fourier se encuentra en la tesis de Vargas Martínez [Vargas-Martínez 2008].

2.1 Análisis Wavelet

Una wavelet es una forma de onda de duración limitada efectivamente que tiene un valor promedio de cero.

Comparando los wavelet con ondas seno (Figura 2.1), que son la base del análisis de Fourier, los senos no tienen una duración limitada, se extienden desde $-\infty$ hasta ∞. Los senos son suaves y predecibles, debido a que son periódicos, los wavelets tienden a ser irregulares y asimétricos [Misiti et al. 2010].



Onda seno (base de análisis de Fourier)

Wavelet (db10)

Figura 2.1 - Onda seno y wavelet a simple vista se observa las diferencias entre ellas

El análisis de Fourier consiste en descomponer una señal en ondas seno de varias frecuencias. De manera similar, el análisis wavelet descompone una señal en versiones escaladas y recorridas de la wavelet original.

Los wavelets son una extensión del análisis de Fourier. Así como con la transformada de Fourier, el punto de los wavelets no son los mismos wavelets; ellos son métodos para un fin. La meta es el cambiar la información de una señal en números—coeficientes-- que puedan ser manipulados, almacenados, transmitidos, analizados, o usados para reconstruir la señal original. [Burrus et al. 1997]

El acercamiento básico es el mismo. Los coeficientes dicen en que forma la función de análisis necesita ser modificada para poder reconstruir la señal. Uno puede literalmente construir la señal al agregar juntos wavelets de diferentes tamaños, en diferentes posiciones, así como uno construye una señal al agregar senos y cosenos. La técnica para la computación de los coeficientes es la misma:

La señal y la función de análisis son multiplicadas juntas, y la integral del producto es computada; Comprimir y estirar los wavelets para cambiar su frecuencia cambia todo. Las wavelets automáticamente se adaptan a los diferentes componentes de la señal, usando una pequeña ventana para mirar a componentes breves de alta frecuencia y una ventana grande para mirar componentes largos de baja frecuencia. Este procedimiento es llamado multiresolución; la señal es estudiada a una resolución “tosca” para tener un panorama general y a resoluciones más altas para visualizar detalles más finos. Las Wavelets han sido llamadas un “microscopio matemático”; comprimiendo wavelets incrementa el aumento del microscopio, permitiendo ver más de cerca pequeños detalles en la señal.

Típicamente, como se muestra en la Figura 2.2, cinco resoluciones diferentes son usadas, cada una dos veces más fina que la resolución previa. Resolución sugiere el número

de wavelets usados. La escala evoca el hecho de que cambiando el tamaño de la wavelet cambia el tamaño de los componentes que uno puede ver. La naturaleza peculiar de los wavelets – El resultado de su tamaño constante- es que la resolución, escala y frecuencia cambian todas a la vez.

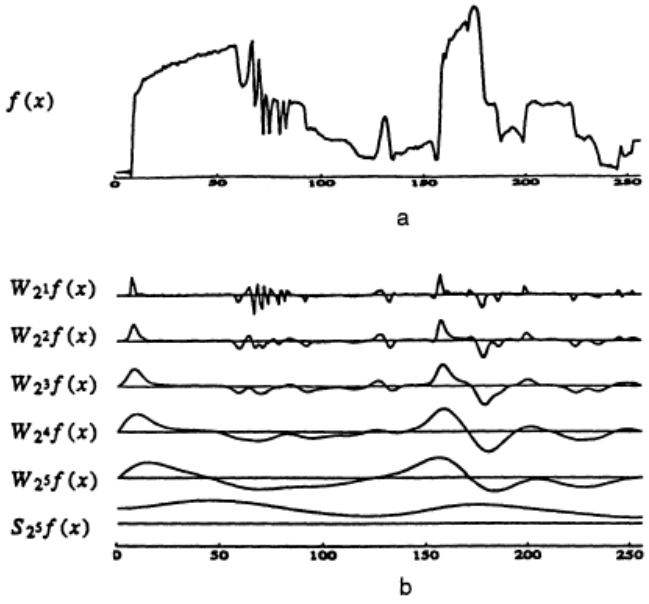


Figura 2.2 - Un ejemplo de una transformada wavelet (a) La señal original. (b) La transformada wavelet de la señal, sobre 5 escalas, difiriendo por un factor de 2. La resolución final, que da los detalles más pequeños, está arriba. En la parte baja de la grafica estan las bajas frecuencias restantes.

El análisis Wavelet representa una técnica de ventaneo con regiones de tamaño variable. El análisis Wavelet permite el uso de intervalos de tiempo largo donde se espera información de baja frecuencia más precisa y regiones cortas donde se espera información de alta frecuencia (Figura 2.3)[Misiti et al. 2010].

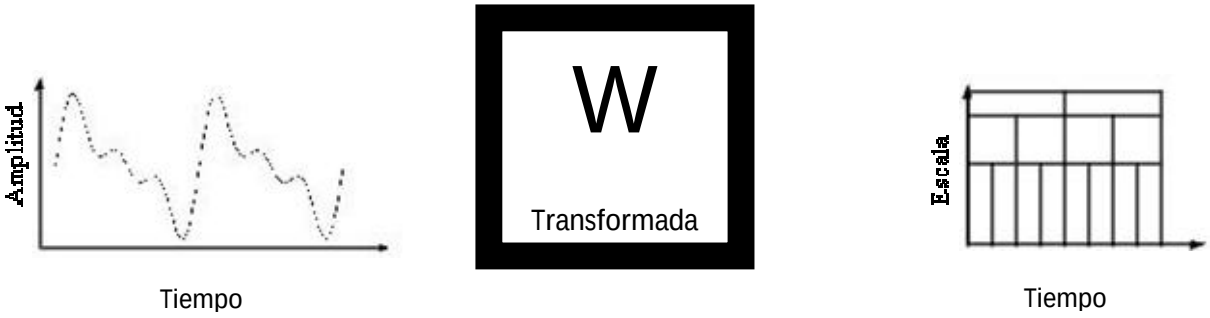


Figura 2.3 - Transformada wavelet aplicada a una señal

A continuación se puede observar una comparación entre las vistas de una señal basada en tiempo, en frecuencia, y en SFTF (Figura 2.4).

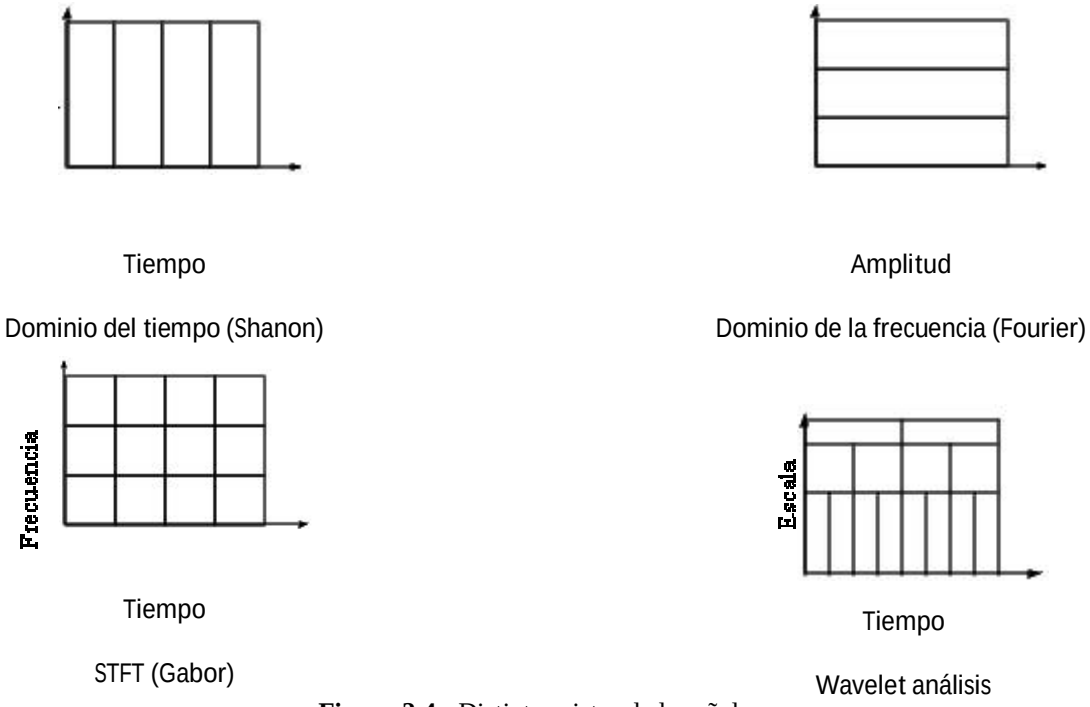


Figura 2.4 - Distintas vistas de la señal

Se puede observar que el análisis wavelet utiliza una región de tiempo-escala, en lugar de una región de tiempo-frecuencia.

2.1.1 Transformada Wavelet

Los siguientes conceptos han sido extraídos del libtro *Elements of Wavelet for Engineers and Scientists* **[Mix and Olejniczak, 2003]** donde trata de una manera precisa los conceptos necesarios para entender la transformada wavelet.

Existen dos formas para determinar la transformada wavelet. Muchas transformadas solo tienen una forma, una fórmula. Por ejemplo, la transformada de Laplace de una función de tiempo $v(t)$ es calculada por la fórmula

$$v(s) = \int_{-\infty}^{-\infty} v(t)e^{-st}dt$$

Existe una fórmula para la transformada wavelet.

$$F_{\omega}(\omega,t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(u)e^{-j\omega u} w(u-t)du$$

La segunda forma es la manera en que la transformada wavelet es calculada en la práctica.

Se empieza con muestras de una señal de tiempo-continuo sobre algún intervalo. Digamos que el intervalo es suficientemente grande para incluir 1024 muestras. Proporcionar esta señal de tiempo-discreta a dos filtros digitales en paralelo. La Figura 2.5 muestra el proceso, donde H_0 es un filtro pasa-bajo y H_1 es un filtro pasa-alto. La entrada es $s(n)$ y $c(n)$ junto con $d(n)$ son los términos de salida. El símbolo $(\downarrow 2)$ representa el muestreo hacia abajo por 2, que significa que se elimina cualquier otra muestra. Se mantiene la 1era, 3era, 5ta, … muestras y se eliminan la 2da, 4ta, 6ta, … . En este esquema $c(n)$ contiene 512 muestras, así como $d(n)$.

Los 512 términos $d(n)$ representan parte de la transformada. La otra mitad es extraída de $c(n)$ al repetir la operación de filtrado de muestreo hacia abajo. La Figura 2.6 muestra exactamente lo que esto significa.

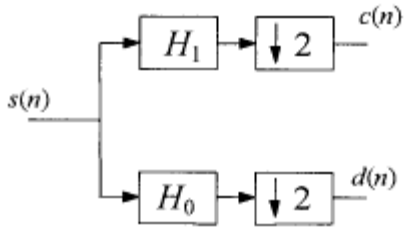


Figura 2.5 - La primer fase de la transformada.

La transformada wavelet contiene el mismo número de términos que la señal, 1024 en este ejemplo. Los términos en la transformada consisten de d_9 (512 muestras), d_8 (256 muestras), d_7 (128 muestras), d_6 (64 muestras), d_5 (32 muestras), d_4 (16 muestras), d_3 (8 muestras), d_2 (4 muestras), d_1 (2 muestras), más un término en c_0.

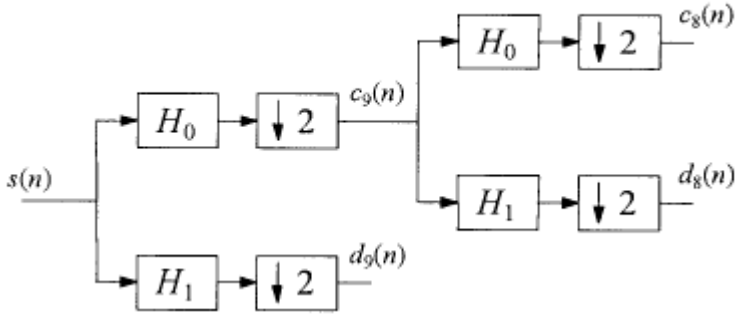


Figura 2.6 - Las primeras dos fases de la transformada.

Para encontrar la transformada inversa, se revierte el proceso. Se comienza con los dos términos c_0 y d_0. Proporcionar estos dos filtros digitales en paralelo, como se muestra en la Fig. 2.7. El simbolo ($\uparrow 2$) representa un muestreo hacia arriba por 2, lo que significa que se insertan ceros entre muestras. Empezando con una señal de longitud 4, dada por {1, 2, -1, 3}, la señal resultante muestreada hacia arriba es {1, 0, 2, 0, -1, 0, 3, 0}, la cual tiene longitud 8. Inicialmente, hay una muestra en la entrada de cada rama filtro de muestra hacia

arriba, la salida contendrá dos muestras. Para los filtros apropiados H_2 y H_3, la salida del primer **verano** será $c_1(n)$

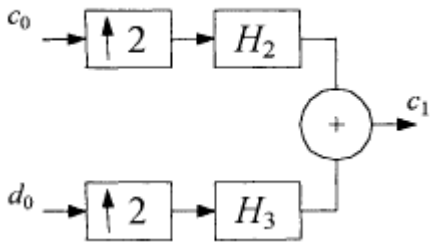


Figura 2.7 - La transformada inversa.

Se ha hablado poco sobre los filtros en el proceso, a continuación se hablará sobre ellos.

Considerar la relación entre H_0 y H_1. La Figura 2.8 muestra la situación que no funcionaría. Entre ellos, los dos filtros deben pasar todas las frecuencias en el rango $0 \le \Omega < 2\pi$. En la Fig. 2.8, las frecuencias entre $2\pi / 3$ y π son bloqueadas de pasar a través de cualquiera de los dos filtros. Estos filtros no pueden preservar información.

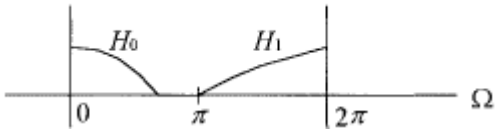


Figura 2.8 - Dos reacciones de frecuencia.

La Figura 2.9 muestra un banco de filtro que representa la relación entre los cuatro filtros sin las operaciones $(\downarrow 2)$ y $(\uparrow 2)$. Considere la relación entre H_0 y H_2, o la rama superior en la Fig. 2.9 Si la salida *y* contiene toda la información en la entrada *v*, entonces H_2 debe ser un filtro pasa-bajas. Supongamos lo contrario, que H_0 es un filtro pasa-bajas y H_2 es pasa-altas, entonces todos los componentes de frecuencia serán bloqueados de pasar a través de la rama superior. De manera similar, si H_1 es pasa-altas, entonces H_3 debe ser también pasa-altas.

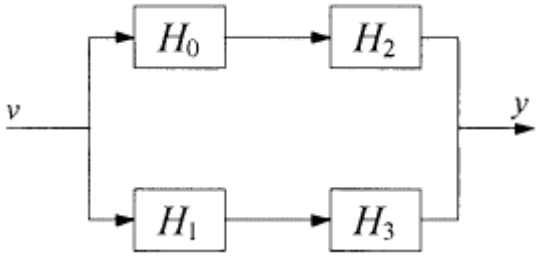


Figura 2.9 - Banco de filtros con los cuatro filtros.

La transformada wavelet difiere de otras transformadas en dos formas.

La base de la transformada de Laplace consiste en un conjunto de funciones exponenciales$\{e^{st}\}$.La base de la transformada z es tambien un conjunto de funciones exponenciales,$\{z^n\}$. Cada forma de la transformada de Fourier tiene su propia base. Si seguimos la costumbre de asociar una transformada particular con su base, existen muchas transformadas wavelet.

Una segunda diferencia ocurre en el calculo de la transformada. Muchas transformadas usan el producto punto entre una onda v(*t*) y las funciones base. Esto funciona para la transformada wavelet si una expersion analitica para las funciones base wavelet es conocida. Pero no hay fórmula para la mayoría de los wavelets, asi que otro método debe ser usado.

2.1.2 Funciones de escalamiento y wavelets.

Un vector puede ser otras cosas ademas de una magnitud dirigida o un matriz de *n* x 1. Se utiliza en el texto los espacios vectoriales, a continuación se definirá:

Definicion 2.1. Un espacio vectorial es un conjunto V=$\{v_i\}$ junto con un campo de escalares $A = \{a_i\}$ que tiene las dos operaciones siguientes y siete propiedades.

1. Dos vectores se pueden añadir para obtener un tercer vector. Este mecanismo combina dos vectores para producir un tercero,
2. Un escalar puede multiplicar un vector para obtener otro vector. Este mecanismo combina un escalar con un vector para producir otro vector.

Usando estas dos operaciones, adicion de vectores y multiplicacion escalar, las siguientes propiedades se deben manter para todo$v_i \in V$y todo $a_i \in A$.

(1) $v_1 + v_2 = v_2 + v_1$.

(2) $(v_1 + v_2) + v_3 = v_1 + (v_2 + v_3)$.

(3) $a_1(v_1 + v_2) = a_1v_1 + a_1v_2$.

(4) $(a_1 + a_2)v_1 = a_1v_1 + a_2v_2$.

(5) $(a_1a_2)v_1 = a_1(a_2v_1)$.

(6) $1v_1 = v_1$

(7) Existe un unico vectorv_0, llamado el vector zero, tal que para todos los vectores v_1 tenemos que $0v_1 = v_0$.

Ahora pensemos en ondas. Para ser mas especificos, consideremos señales de energía de tiempo-continuo. Cualesquiera dos de ellas se añadiran para obetener una tercera señal de energía. Multiplicando por escalares cambia la amplitud de la señal. Y las señales de energia satisfacen las siete propiedades, asi que la señal de energia satisface los requerimientos para ser llamado un espacio vectorial. Esto significa que cada señal de energia es un vector. Ademas, una señal de energia es igual de buena a un vector como una magnitud dirigida o un vector componente.

En una manera similar, el conjunto de todos las señales de potencia de tiempo-continuo forma otro espacio vectorial, asi como el conjunto de todoas las señales de energia de tiempo-discretas, o el conjunto de todas las señales de potencia de tiempo-discretas, Y

existen muchos otros espacios vectoriales. He aqui algunos ejemplos:

Ejemplo 2.1. La figura 2.10 muetra una onda constante por tramos, definida en le intervalo [0, 1]. Existen cuatro segmentos constantes. [0, ¼], [¼, ½], [½, ¾] y [¾, 1]. Sea *V* el conjunto de todas las ondas de ese tipo. Definir el producto interno como

$$\langle v_1 | v_2 \rangle \int_0^1 v_1(t) v_2(t) dt$$

(a) Demostrar que *V* es un espacio vectorial.

(b) Encontrar la base para *V*.

(c) Determinar la dimension de *V*.

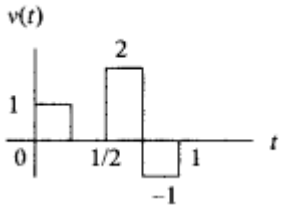


Figura 2.10 - Una función constante por tramos (un vector).

Solución: (a) Este conjunto satisface los dos operadores en la Definicion 1, porque podemos añadir cualesquiera dos funciones constantes por tramos y obtener otra función constante por tramos, y podemos multiplicar cualquier función por un escalar y obtener otra función constante por tramos. Este conjunto tambien satisface las siete propiedades.

(b) La Figura 2.11 muestra una posible base, y la Fig. 2.12 muestra otra. Estas dos bases tienen ciertas propiedades buenas. La base $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$en la Fig. 2.11 es ortogonal. Esto significa que el producto interno es zero para cualquiera de dos vectores base diferentes. Sin embargo, esta base no es ortonormal, porque el producto interno de $\alpha_i$consigo mismo no es 1 para todo *i*. La base en la Fig. 2.12 es ortonormal, porque el producto interno entre dos vectores diferentes es cero, mientras que el producto interno de cada base consigo misma es 1.

(c) La dimension de *V* es 4 porque existen cuatro vectores base.

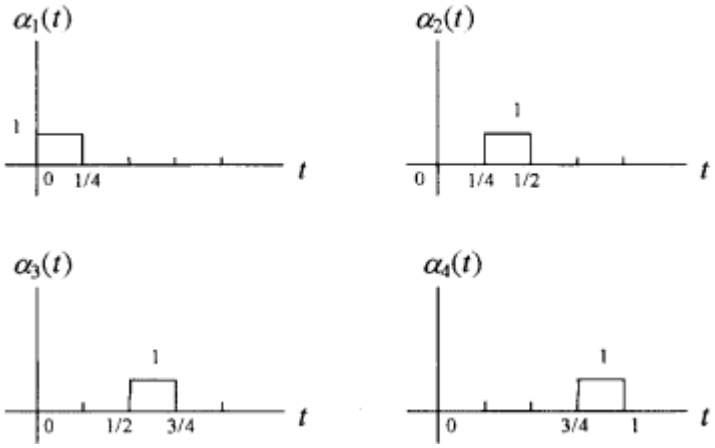


Figura 2.11 - Una posible base para *V*.

Ejemplo 2.2. Expresar *v*(*t*) en la Fig. 2.10 por sus componentes con respecto a (a) la base α y (b) la base β.

Solucion: (a) Compare la señal en la Fig. 2.10 a los cuatro vectores base en α y ver por inspección que *v*(*t*) puede ser expresado como una combinacion linear de sus vectores base.

$$v(t) = \alpha_1(t) + 2\alpha_3(t) - \alpha_4(t)$$

$$[v]_a = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}$$

Por lo tanto, el vector componente que representa v(t) con respecto a la base α está dado por

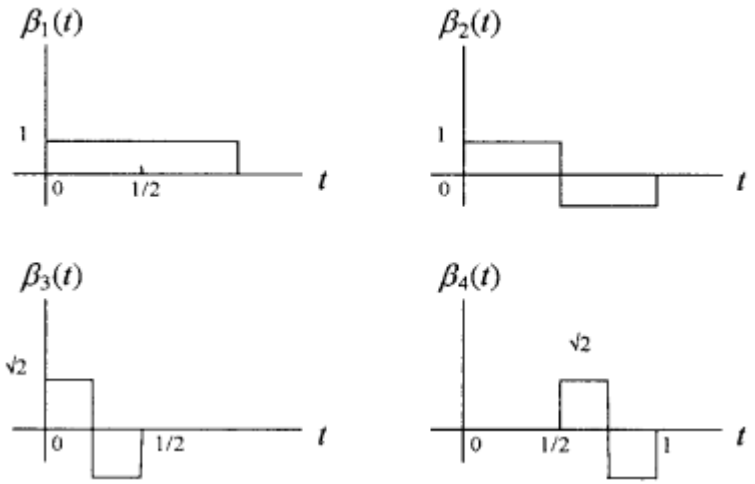


Figura 2.12 - Una base ortonormal para *V*.

(b) Aqui la solución no es tan obvia, requiere un acercamiento sistematico. El vector componente $[v]_\beta$ consiste de los cuatro componentes $b_1, b_2, b_3 y b_4$ de la relación

$$v(t) = b_1\beta_1(t) + b_2\beta_2(t) + b_3\beta_3(t) + b_4\beta_4(t) \qquad (2.1)$$

la cual representa v(t) como una combinación lineal de los vectores base en β. Existen cuatro incógnitas, b_1, b_2, b_3, b_4. Para desarrollar cuatro ecuaciones independientes, tomar el producto interno de cada término en esta ecuación con los cuatro vectores base

$$\langle \beta_1 | v \rangle = b_1 \langle \beta_i | \beta_1 \rangle + b_2 \langle \beta_i | \beta_2 \rangle + b_3 \langle \beta_i | \beta_3 \rangle + b_4 \langle \beta_i | \beta_4 \rangle \quad \text{para } i = 1,2,3,4$$

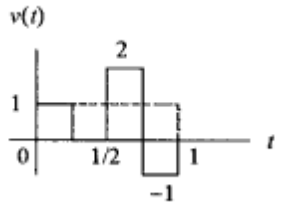


Figura 2.13 - Las ondas $\boldsymbol{\beta_1(t)}$ y $\boldsymbol{v(t)}$.

La figura 2.13 muestra las ondas $\beta_1(t)$ y $v(t)$. Multiplicar las dos funciones y encontrar el área bajo el producto. Este producto interno es ¼ [1 + 2 - 1] = ½. En una forma similar, los otros coeficientes dan el vector componente

La ecuacion 2.1 resulta en $v(t) = \frac{1}{2}\beta_1(t) + \frac{\sqrt{2}}{4}\beta_3(t) + \frac{3\sqrt{2}}{4}\beta_4(t).$

Aqui la nueva idea. Pensar en los vectores coeficientes como la transformada de $v(t)$. De este modo, $[v]_\alpha$ es la transformada de $v(t)$ con respecto a la base α, y $[v]_\beta$ es la transformada de v(t) con respecto a la base β. Escribirlo como

$$v(t) \xleftrightarrow{a} \{1, 0, 2, -1\} \qquad (2.2)$$

$$v(t) \xleftrightarrow{b} \{\tfrac{1}{2}, 0, \tfrac{\sqrt{2}}{4}, \tfrac{3\sqrt{2}}{4}\} \qquad (2.3)$$

La ecuacion 2.3 es la transformada wavelet Haar de $v(t)$. Las funciones Haar con ondas constantes por tramos, y las usaremos para ilustrar las propiedades generales de los wavelets antes de expander la discusión para incluir otras wavelets.

Es legitimo llamar a esto una transformada. Una transformada es una relación entre dos conjuntos de funciones(vectores). En las aplicaciones el dominio consiste en funciones de tiempo. Los vectores de salida en el codominio son llamadas la transformada, y en nuestro caso estas son las transformadas wavelet de la entrada de funciones de tiempo

La transformada wavelet de una señal $v(t)$ es el conjunto de proyecciones de $v(t)$ a una base. Esta base es una familia de funciones que son dilataciones y traslaciones normalizadas de un wavelet prototipo $\psi(t)$. Ademas, otro conjunto de funciones $\varphi(t)$, llamadas las funciones de escalamiento, estan involucradas. Existe una relacion entre estos dos conjuntos de funciones que involucran el complemento ortogonal de un conjunto.

Sea V_1 un espacio vectorial, y sea V_0 un subconjunto de V_1, escrito $V_0 \subset V_1$. Notese que V_0 puede o no ser un espacio vectorial. Para que V_0 sea un subespacio, debe satisfacer todas las propiedades en la definición para espacio vectorial dada en la definicion 2.1.

Definicion 2.2. Si V_0 es un subconjunto de un espacio vectorial V_1, entonces definimos el complemento ortogonal de V_0, denotado $V_0{}^{\perp}$ y pronunciado V_0-perp, como el conjunto de todos los vectores V_1 los cuales son ortogonales a cada elemento de V_0; esto es,

$$V_0^{\perp} = \{w \in V_1 \,|\, \langle w | v \rangle = 0 \qquad \text{para todo } v \in V_0$$

Ejemplo 2.3 Sea $V_1 = R^2$, el conjunto de todas las matrices de 2 x 1. Sea $V_0 = \left\{\begin{bmatrix}1\\1\end{bmatrix}\right\}$, un conjunto que consiste de una sola matriz. Definir el producto punto por $\langle w \mid v \rangle = w^{(t)}v$. Encontrar $V_0{}^{\perp}$ y mostrar que es un subconjunto de V_1.

Solución: Cada vector w que es ortogonal a $v = \begin{bmatrix}1\\1\end{bmatrix}$ debe tener la forma $w = a\begin{bmatrix}1\\-1\end{bmatrix}$ significando que el complemento ortogonal esta dado por $V_0{}^{\perp} = \begin{bmatrix}a\\-a\end{bmatrix}$. Este conjunto es un espacio vectorial porque $0_v \in V_0^{\perp}$ y todos los vectores de la forma $a_1 w_1 + a_2 w_2$ son tambien elementos de $V_0^{\perp}$.

Es facil demostrar que si V_0 es solo un conjunto, $V_0{}^{\perp}$ es un subespacio de V_1. V_0 siempre sera un subespacio de V_1. Esto permite descomponer vectores dentro de V_1 en la suma de dos espacios vectoriales. Esta composicion es llamada *suma directa*.

Definicion 2.3. Si V_0 es un subespacion finito-dimensional del espacio de producto interno V_1, cada vector $y \in V_1$ puede ser descompuesto unicamente como $y = v + w$, donde $v \in V_0$ y $w \in V_0{}^{\perp}$. Podemos decir que V_1 es la *suma directa* de V_0 y $V_0{}^{\perp}$ y escribimos

$$V_1 = V_0 \oplus V_0^{\perp}$$

He aqui un ejemplo para fijar estas ideas.

Ejemplo 2.4 Sea V_1 el conjunto de todas las funciones constantes por tramos en los intervalos $0 \le t < 1/2$y $1/2 \le t < 1$, como se muestra en la Fig. 2.14. Esto es, si y(t) es miembro de V_1 entonces:

$$y(t) = \begin{cases} k_1, 0 \le t < \frac{1}{2} \\ k_2, \frac{1}{2} \le t < 1 \end{cases} \qquad (2.4)$$

donde k_1 y k_2 son constantes.

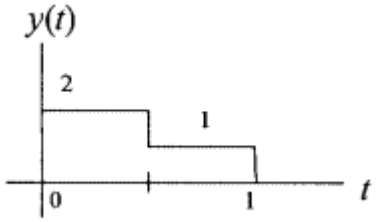


Figura 2.14 - La función constante por tramos *y*(*t*).

Sea V_0 el conjunto de todos las funciones de valor constante (Haar) en el intervalo$0 \le t < 1$. La figura 2.15 muestra una funcion de este tipo,

$$\varphi_{00} = 1, 0 \le t < 1.$$

Otras funciones pueden tener valores de 1.5, -1, 2, y asi en adelante. De hecho φ_{00}, es una base para V_0 porque cualquier miembro de V_0es simplemente un multiplo de φ_{00} .

Definir el producto interno como

$$\langle y_1 | y_2 \rangle = \int_0^1 y_1(t) y_2(t) dt \qquad (2.5)$$

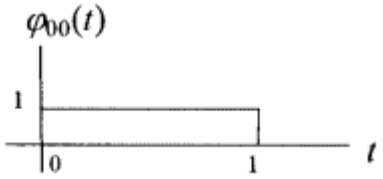


Figura 2.15 - La función $\varphi_{00}(t)$

V_1 y V_0 son dos espacios vectoriales, con $V_0 \subset V_1$. Encontrar $W_0 = V_0^{\perp}$, el complemento ortogonal de V_0.

Cualquier función con simetria irregular alrededor del punto t = ½ es ortogonal a cada vector en V_0. Sin embargo, la condición de que $V_0^{\perp}$ sea un subconjunto de V_1 debe ser satisfecha, asi que la busqueda debe ser restringida a funciones constantes por tramos. La figura 2.16 muestra una función $\psi_{00}(t)$ que satisface ambas propiedades. Es ortogonal a cada función en V_0 y es un miembro de V_1. Todos los multiplos de forman un espacio vectorial $W_0 = V_0^{\perp}$

Despues, notar que los dos vectores $\varphi_{00}(t)$ y $\psi_{00}(t)$ constituyen la base para V_1. Entonces, podemos expresar cada onda en V_1como una combinacion lineal de esos dos vectores en una y solo una manera. El siguiente ejemplo ilustra esto para la onda en la Fig. 2.14.

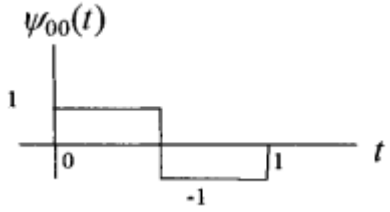


Figura 2.16 - La función $\psi_{00}(t)$

Ejemplo 2.5 Expresar $y(t)$ en la Fig. 2.14 como una combinacion linear de las funciones base $\varphi_{00}(t)$ y $\psi_{00}(t)$ en las Figs. 2.15 y 2.16. Asumir que $k_1 = 2$ y $k_2 = 1$.

Solucion: Escribir

$$y(t) = a_1\varphi_{00}(t) + a_2\psi_{00}(t) \qquad (2.6)$$

Para evaluar $a_{1,}$ tomar el producto interno de cada termino en esta ecuacion con φ_{00}

$$\langle y(t)|\varphi_{00}(t)\rangle = a_1\langle\varphi_{00}(t)|\varphi_{00}(t)\rangle + a_2\langle\psi_{00}|\varphi_{00}(t)\rangle \qquad (2.7)$$

El primer producto interno a la derecha es 1 porque φ_{00} tiene norma unitaria. El segundo producto interno es cero porque φ_{00} y ψ_{00} son ortogonales. Esto da como resultado

$$a_1 = \langle y(t)|\varphi_{00}(t)\rangle = \int_0^{1/2} 2dt + \int_{1/2}^{1} 1dt = \tfrac{3}{2}$$

Similarmente, tomar el producto interno de cada término en la ecuación 2.6 con ψ_{00} para obtener

$$a_2 = \langle y(t)|\psi_{00}(t)\rangle = \int_0^{1/2} 2dt - \int_{1/2}^{1} 1dt = \tfrac{1}{2}$$

dando como resultado

$$y(t) = \tfrac{3}{2}\varphi_{00}(t) + \tfrac{1}{2}\psi_{00}(t)$$

Ejemplo 2.6 Sea V_1el conjunto de todoas las señales de tiempo discretas de tamaño 2. Por ejemplo, $v(n) = \{2, -1\}$es una señal en el conjunto V_1. Sea V_0el subconjunto de V_1 donde dos valores de señales son iguales. Por lo tanto $\varphi_{00}(n) = \{1,1\}$ es un miembro de V_0.

(a) Demostrar que V_0 es un subespacio de V_1

(b) Encontrar el complemento ortogonal $W_0 = V_0^{\perp}$. Seleccionar como $\psi_{00}(n)$ un miembro de W_0.

(c) Demostrar que $\varphi_{00}(n)$ y $\psi_{00}(n)$ constituyen una base para V_1

Solución: (a) V_0 es un subespacio de V_1 porque el vector cero $\{0\ 0\}$ es un miembro de V_0, y combinaciones lineales de vectors en V_0 también estan en V_1. Esto es, si $v_1(n) = \{a, a\}$ y $v_2(n) = \{b, b\}$, entonces $v_1(n) + v_2(n)$ también tienen el mismo valor para los dos valores de señal y es entonces un miembro de V_0. V_0 automáticamente hereda las otras propiedades de un espacio vectorial dado que V_1 es un espacio vectorial y V_0 es un subconjunto de V_1.

(b) El complemento ortogonal $W_0 = V_0^{\perp}$ es el conjunto de todos los vectores en V_1 que son ortogonales a cada vector en V_0. Esto significa que existen dos requerimientos para que una señal $\psi_{00}(n)$ sea miembro de W_0 (1) $\psi_{00}(n)$ debe estar en V_1: (2) $\psi_{00}(n)$ debe ser ortogonal para todos los miembros de V_0. Por lo tanto, $\psi_{00}(n)$ debe tener una longitud de 2 y el producto interno entre $\psi_{00}(n)$ y cada miembro de V_0 debe ser cero. Ambas condiciones son satisfechas por $\psi_{00}(n) = \{1, -1\}$.

(c) $\phi_{00}(n)$ y $\psi_{00}(n)$ constituyen una base para V_1 porque cada señal de longitud 2 puede ser expresada como una combinación lineal de dos señales. Esto es, para una señal arbitraria $v(n)$ de longitud 2, se pueden encontrar escalares a_1 y a_2 de tal forma que

$$v(n) = a_1\phi_{00}(n) + a_2\psi_{00}(n)$$

Nótese que esto también demuestra que cada vector en V_1 esta en V_0 o W_0, Significando:

$$V_1 = V_0 \oplus W_0.$$

Ejemplo 2.7. Encontrar los coeficientes para apoximar $y(t) = 1 + \sin(2\pi t)$ por las funciones $\phi_{00}(n)$ y $\psi_{00}(n)$ en las Figuras 2.15 y 2.16.

Solución:

$$a_1 = \langle y(t) \mid \varphi_{00}(t) \rangle = \int_0^1 [1 + \sin(2\pi t)]\, dt = 1$$

$$a_2 = \langle y(t) \mid \psi_{00}(t) \rangle = \int_0^{1/2} [1 + \sin(2\pi t)]\, dt - \int_{1/2}^1 [1 + \sin(2\pi t)]\, dt = \frac{2}{\pi}$$

Por lo tanto,

$$y(t) \approx \varphi_{00}(t) + (2/\pi)\psi_{00}(t)$$

Aquí un resumen de algunas notaciones relativas a las ondas. Sea $v(t)$ la onda en la Figura 2.17a. Entonces $v(2t)$, $v[2(t-1)]$ y $v(2t-1)$ tienen las formas mostradas en las Figuras 2.17b, c y d, respectivamente.

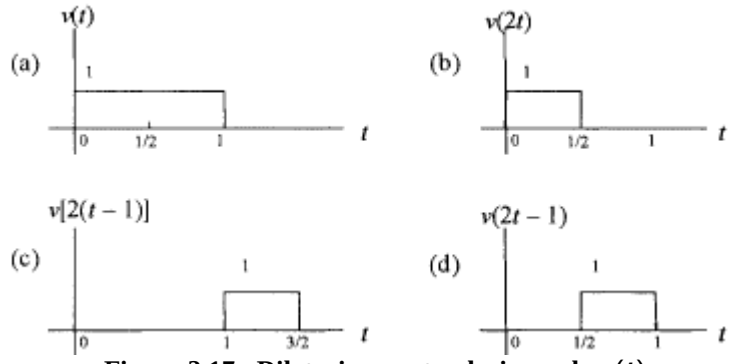


Figura 2.17 - Dilataciones y traslaciones de $v(t)$

La notación es importante para pensar mas claramente, y la notación antigua de $V_1 = V_0 \oplus V_0^{\perp}$ es rara. La notación debe ser arreglada en preparación para expandir los temas de subespacios, sumas directas, y complementos ortogonales. El último ejemplo uso notación adecudada. Sea

$$W_0 = V_0^{\perp}$$

Podemos escribir

$$V_1 = V_0 \oplus W_0.$$

Si $\phi_{00}(t)$ y $\psi_{00}(t)$forman una base para V_1, entonces V_0 consiste en multiplos de $\phi_{00}(t)$,W_0 consite en multiplos de $\psi_{00}(t)$, y V_1consiste en multilpos de $\phi_{00}(2t-k)$, $k = 0,1$. Entonces, no solo V_1 tiene una base que consiste de $\{\phi_{00}(t), \psi_{00}(t)\}$, pero otra base es $\{\phi_{00}(2t)\ y\ \phi_{00}(2t-1)\}$.

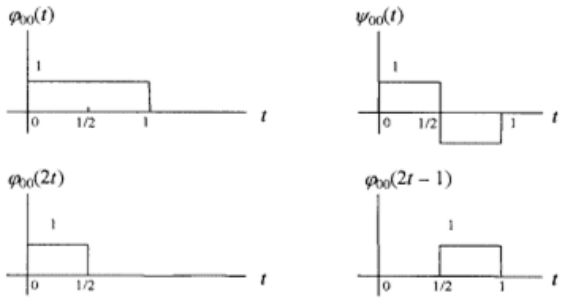


Figura 2.18 - Base 1 y Base 2.

Base 1: $\{\phi_{00}(t), \psi_{00}(t)\}$
Base 2: $\{\phi_{00}(2t), \phi_{00}(2t-1)\}$

Nótese que

$$\phi_{00}(t) = \phi_{00}(2t) + \phi_{00}(2t-1)$$

y

$$\psi_{00}(t) = \phi_{00}(2t) - \phi_{00}(2t-1)$$

Debido que $\{\phi_{00}(t), \psi_{00}(t)\}$ puede ser expresado como una combinación lineal de los $\{\phi_{00}(2t), \phi_{00}(2t-1)\}$ vectores, entonces$\{\phi_{00}(2t), \phi_{00}(2t-1)\}$ debe ser tambien una base de V_1

Para explorar esta segunda base mas a fondo esta base, sea

$$\varphi_{jk}(t) = 2^{j/2}\varphi_{00}(2^j t - k) \qquad (2.8)$$

donde el factor $2^{(j/2)}$ normaliza cada función para tener la unidad energía. Esta notación significa que

$$j=0,\ k=0,\ 2^{0}\varphi_{00}(2^{0}t-0)=\varphi_{00}(t)$$
$$j=1,\ k=0,\ 2^{1/2}\varphi_{00}(2^{1}t-0)=\sqrt{2}\varphi_{00}(2t)$$
$$j=1,\ k=1,\ 2^{1/2}\varphi_{00}(2^{1}t-1)=\sqrt{2}\varphi_{00}(2t-1)$$
$$j=2,\ k=0,\ 2^{1}\varphi_{00}(2^{2}t-0)=2\varphi_{00}(4t)$$
etc.

donde k esta entre 0 y $2^{j}-1$. Idealmente *j* empieza en 0 y va hasta ∞. En la práctica, el límite superior en *j* es determinado por la velocidad de sampleo. Calculos prácticos de los coeficientes wavelet empiezan en el otro final. En lugar de empezar con *j*= 0 y trabajar el camino hacia arriba hacia el máximo valor de *j*, se empieza en la resolución mas fina y trabajar hacia abajo. Como se muestra en la Fig. 2.19 para la base 2,

$$\varphi_{10}(t)=\sqrt{2}\varphi_{00}(2t) \quad \text{y} \quad \varphi_{11}(t)=\sqrt{2}\varphi_{00}(2t-1)$$

Para definir un mayor refinamiento del intervalo $0\le t<1$, sea V_2el conjunto de todas las funciones constantes por tramos en los intervalos $0\le t<\frac{1}{4},\frac{1}{4}\le t<\frac{1}{2},1/2\le t<3/4$, y $3/4\le t<1$.Estas son las funciones ϕ_{2k}mostradas en la Fig. 2.19. De la ecuación 2.8,

$$\varphi_{2k}(t)=2\varphi_{00}(4t-k), \qquad k=0,1,2,3$$

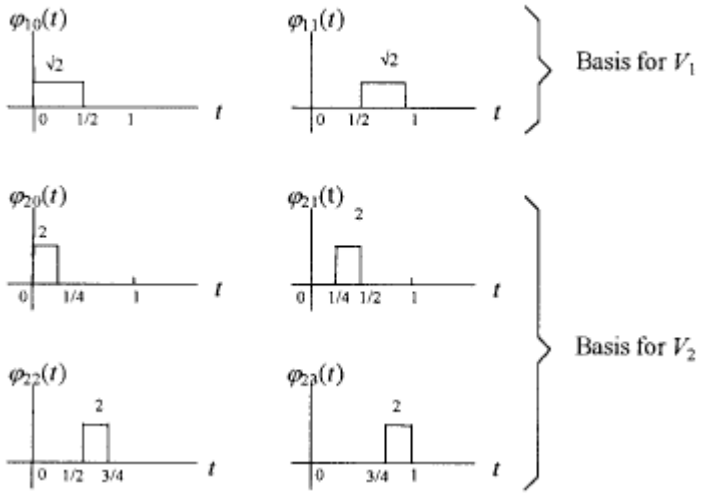


Figura 2.19 - Bases para V_1 y V_2

Las $\phi_{1k}(t)$ funciones forman una base para V_1, y las $\phi_{2k}(t)$funciones forman una base para V_2, y este patron puede continuar. Esto genera una jerarquía de espacios vectoriales definidos por sus funciones base en la Ec.2.8.

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2 \qquad (2.9)$$

Donde L^2 es el conjunto de todas las señales de energía.

Las funciones $\{\phi_{jk}\}$son llamadas funciones de escalamiento, las funciones$\{\psi_{jk}\}$son llamadas wavelets. Las funciones de escalamiento forman una base para V_j. Las wavelets forman una base para $W_j = V_j^{\perp}$. Similar a las funciones de escalamiento, estan definidas por

$$\psi_{jk}(t) = 2^{j/2}\psi_{00}(2^j t - k) \qquad (2.10)$$

La figura 2.20 muestra los wavelets Haar para $j = 1$ y $j = 2$.

Note que una combinación de funciones escalares mas los wavelets forman una base para el siguiente espacio mas alto. Pero otras combinaciones pueden ser usadas: solamente funciones de escalamiento, o funciones de escalamiento mas wavelets, o todos los wavelets a excepción de una función de escalamiento, $\phi_{00}(t)$, a tomar en cuenta para el nivel dc en la onda. Se debe tener cuidado en escoger una base para hacer los vectores mutuamente ortogonales. Esta propiedad, la cual es compartida por las series de Fourier, hace que sea fácil trabajar con la transformada. La transformada wavelet usa $\phi_{00}(t)$ mas todas las $\psi_{jk}(t)$ wavelets. En otras palabras, escoger

$$V_0 = \text{todos los escalares múltiplos de } \varphi_{00}(t)$$
$$V_0^{\perp} = W_0 \cup W_1 \cup W_2 \cup \cdots$$

esto es, $V_0^{\perp}$ consiste en todos los multiplos escalares de $\{\psi_{jk}\}$para todo *j,k*. Esto hace cada vector en la base sea ortogonal a todos los vectores en todas las resoluciones. Para ver como funciona esto, considerar lo siguiente:

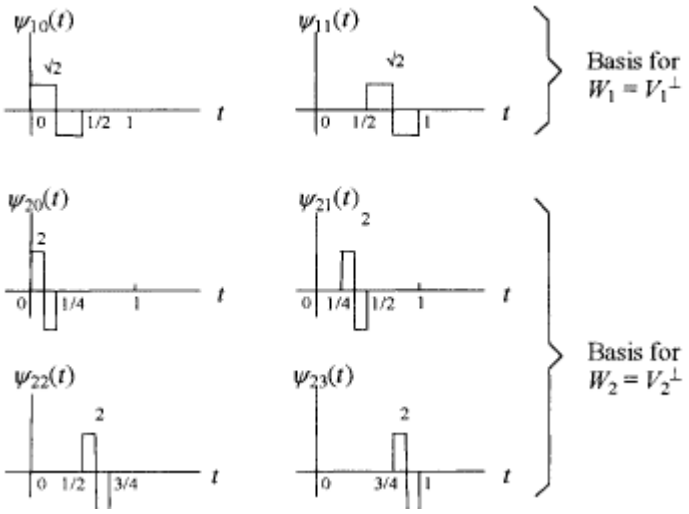


Figura 2.20 - Haar wavelets para *j* = 1 y *j* = 2.

1. Este esquema genera una jeraquía de espacios vectoriales.

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2$$

2. Cada espacio vectorial V_1 puede ser descompuesto en $V_i = V_{i-1} \oplus W_{i-1}$, donde $W_{(i-1)}$ es el complemento ortogonal de $V_{(i-1)}{}^{\perp}$. Esto significa que V_1 tiene una base compuesta de la base para $V_{(i-1)}$combinada con la base para$W_{(i-1)}$. Por ejemplo, V_2 tiene la base

$$\beta_2 = \{\varphi_{00}, \psi_{00}, \psi_{10}, \psi_{11}\}$$

V_3 tiene la bases que consiste de β_2, mas las waveletes que constituyen una base para W_2. De este modo

Estos ejemplos ilustran el esquema para construir las bases para altas resoluciones.

$$\beta_3 = \{\varphi_{00}, \psi_{00}, \psi_{10}, \psi_{11}, \psi_{20}, \psi_{21}, \psi_{22}, \psi_{23}\}$$

2.1.3 Transformada Wavelet discreta.

Existen tres transformadas wavelet comunmente definidas, asi como existen cuatro transformadas de Fourier.

1. La transformada discreta wavelet (DWT) mapea funciones de tiempo-continuas a un conjunto de números, de forma parecerida como la serie de Fourier mapea señales de tiempo-continuas a una conjunto de números.

2. Otra forma de transformada wavelet mapea señales de tiempo-continuas en funciones continuas, como la transformada de Fourier para señales de energía de tiempo-continuas.

3. La tercer forma mapea señales de tiempo-discretas en un conjunto de números, y de esta forma es analoga a la DTFS.

Esta sección se concentra en la DWT. Mapea funciones de tiempo-continuas en números. Las siguientes ecuaciones estan dadas por

$$\begin{aligned} &c_{jk}\left\langle v(t)\middle|\varphi_{jk}(t)\right\rangle \\ &d_{jk}\left\langle v(t)\middle|\psi_{jk}(t)\right\rangle \end{aligned} \qquad (2.11)$$

La ecuación inversa esta dada por

$$v(t)=\sum_{k=-\infty}^{\infty} c_{Jk}\varphi_{Jk}(t)+\sum_{j=J}^{\infty}\sum_{k=-\infty}^{\infty} d_{jk}\psi_{jk}(t) \qquad (2.12)$$

donde J es el índice de comienzo. Las ecuaciones de análisis descomponen (analizan) una onda dada $v(t)$en sus componentes constituyentes c_{jk}y d_{jk}. La Figura 2.21a muestra esta operación. La ecuación de sintesis construye (sintetiza) la función $v(t)$de sus componentes c_{jk} y d_{jk}, la Figura 2.21b muestra esta operación.

Ejemplo 2.8. Encontrar la transformada de la función en la Fig. 2.22. Usar las cuatro funciones en la Fig. 2.23 para la base.

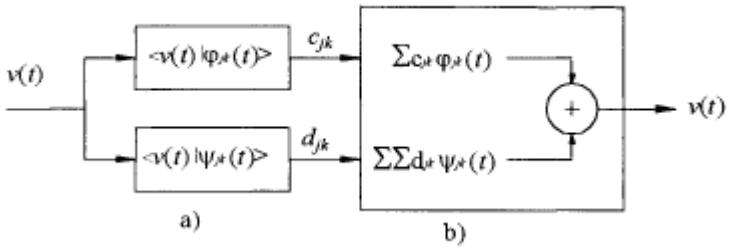


Figura 2.21 - Transformada wavelet hacia adelante e inversa.

Solución: Las ecuaciones (Ec. 2.8) hacia enfrente (análisis), resultan en

$$c_{00} = \langle v(t) \mid \varphi_{00}(t) \rangle = \frac{1}{4} + \frac{2}{4} - \frac{1}{4} = \frac{1}{2}$$

$$d_{00} = \langle v(t) \mid \psi_{00}(t) \rangle = \frac{1}{4} - \frac{2}{4} + \frac{1}{4} = 0$$

$$d_{10} = \langle v(t) \mid \psi_{10}(t) \rangle = \frac{\sqrt{2}}{4}$$

$$d_{11} = \langle v(t) \mid \psi_{11}(t) \rangle = \frac{2\sqrt{2}}{4} + \frac{\sqrt{2}}{4} = \frac{3\sqrt{2}}{4}$$

dando

$$v(t) = \frac{1}{2}\varphi_{00}(t) + \frac{\sqrt{2}}{4}\psi_{10}(t) + \frac{3\sqrt{2}}{4}\psi_{11}(t)$$

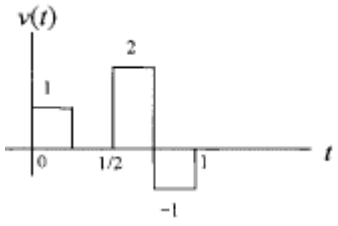


Figura 2.22 - Función $v(t)$

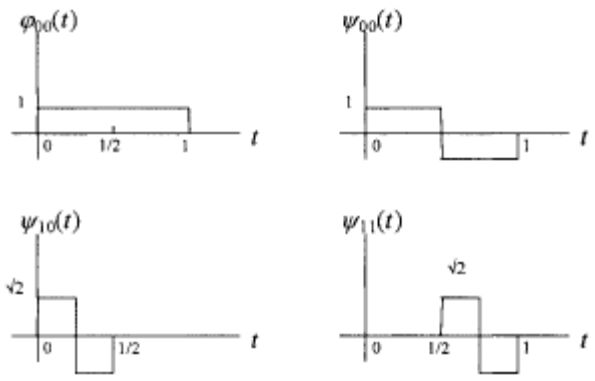


Figura 2.23 - Funciones base para el ejemplo 2.8

Ejemplo 2.9. Calcule la función de escalamiento y coeficientes wavelet a nivel 2 para la función $v(t) = 1 + \sin(2\pi t)$.Usar el sistema wavelet Haar en el ejemplo 2.8.

Solución: La figura 2.24 muestra la funcion de escalamiento y wavelet madre. Los coeficientes de nivel zero estan dados por:

$$c_{00} = \langle v(t) \mid \varphi_{00}(t) \rangle = \int_0^1 [1 + \sin(2\pi t)]\, dt = 1$$
$$d_{00} = \langle v(t) \mid \psi_{00}(t) \rangle = \int_0^{1/2} [1 + \sin(2\pi t)]\, dt - \int_{1/2}^1 [1 + \sin(2\pi t)]\, dt = 0.6366$$

La Figura 2.25 muestra el nivel 1 de escalamiento y funciones wavelet. Su producto interno con $v(t)$ resulta en

$$c_{10} = \langle v(t) \mid \varphi_{10}(t) \rangle = \int_0^{1/2} \sqrt{2}\,[1 + \sin(2\pi t)]\, dt = 1.1573$$
$$c_{11} = \langle v(t) \mid \varphi_{11}(t) \rangle = \int_{1/2}^1 \sqrt{2}\,[1 + \sin(2\pi t)]\, dt = 0.2570$$
$$d_{10} = \langle v(t) \mid \psi_{10}(t) \rangle = \int_0^{1/4} \sqrt{2}\,[1 + \sin(2\pi t)]\, dt - \int_{1/4}^{1/2} \sqrt{2}\,[1 + \sin(2\pi t)]\, dt = 0$$
$$d_{11} = \langle v(t) \mid \psi_{11}(t) \rangle = \int_{1/2}^{3/4} \sqrt{2}\,[1 + \sin(2\pi t)]\, dt - \int_{3/4}^1 \sqrt{2}\,[1 + \sin(2\pi t)]\, dt = 0$$

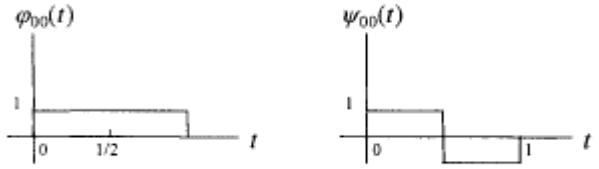


Figura 2.24 - Función Haar de escalamiento y wavelet madre.

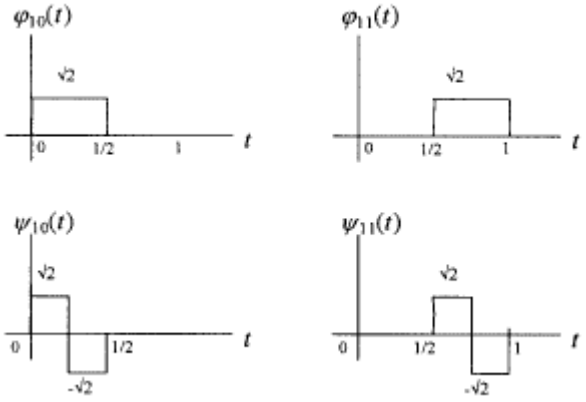


Figura 2.25 - Funciones de escalamiento y wavelet nivel 1.

De una manera similar, los coeficientes de segundo nivel estan dados por:

$$c_2 = \begin{bmatrix} c_{20} \\ c_{21} \\ c_{22} \\ c_{23} \end{bmatrix} = \begin{bmatrix} 0.8183 \\ 0.8183 \\ 0.1817 \\ 0.1817 \end{bmatrix} \qquad \mathrm{d}_2 = \begin{bmatrix} d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \end{bmatrix} = \begin{bmatrix} -0.1319 \\ 0.1319 \\ 0.1319 \\ -0.1319 \end{bmatrix}$$

Resultando en la serie:

$$v(t) = 1 + \sin(2\pi t) \cong c_{00}\varphi_{00}(t) + d_{00}\psi_{00}(t) + d_{10}\psi_{10}(t) + d_{11}\psi_{11}(t) + d_{20}\psi_{20}(t) + d_{21}\psi_{21}(t) + d_{22}\psi_{22}(t) + d_{23}\psi_{23}(t)$$

o

$$v(t) \cong \varphi_{00}(t) + 0.6366\psi_{00}(t) - 0.1313\psi_{20}(t) + 0.1319\psi_{21}(t) + 0.1319\psi_{22}(t) - 0.1319\psi_{23}(t)$$

Antes de proseguir mas con ejemplos de otras wavelets, se mencionaran conceptos que estan implicitos en lo visto hasta el momento.

1. La construccion de wavelets usa funciones de escalamiento $\phi(t)$ que pueden ser escritas como combinaciones lineales de$\phi(2t-k)$, el escalado-medio y versiones trasladadas *k*/2 de $\phi(t)$. Esta combinación lineal es:

$$\varphi(t)=\sum_k h_0(k)\sqrt{2}\varphi(2t-k) \qquad (2.13)$$

donde los términos $\sqrt{2}h_0(k)$ son los coeficientes que relacionan la funcion $\phi(2t-k)$ a $\phi(t)$ esta es llamada la *relación dos-escala* para ϕ, y los coeficientes $\sqrt{2}h_0(k)$ son llamados la secuencia dos-escala de ϕ.

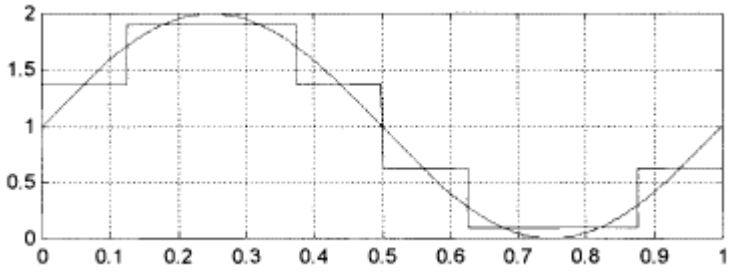


Figura 2.26 - Aproximando $v(t) = 1 + \sin(2\pi t)$con $j = 2$.

2. El subespacio V_0esta abarcado por $\phi_0(t)$y sus traslaciones enteras.

3. Las funciones en Ec. 2.5,

$$\varphi_{jk}(t)=2^{j/2}\varphi_0(2^j t-k)$$

abarcan el espacio V_j. La relación dos-escala en Ec. 2.13 genera una secuencia anidada de

$$\cdots V_{-1}\subset V_0\subset V_1\subset V_2\subset\cdots$$
$$\leftarrow \text{grueso} \quad \text{fino} \rightarrow$$

subespacios:

$$\cdots V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots$$
$$\leftarrow \text{coarser} \qquad \text{finer} \rightarrow$$

Esto es llamado el *análisis multiresolución* (MRA)

4. El espacio V_j y su complemento ortogonal $W_j$son ambos subespacion de $V_{(j+1)}$. Esto es

$$V_{j+1} = V_j \oplus W_j$$

Dada una funcion de escalamiento ϕ en V_j, el principio básico de MRA es que existe otra función ψen W_j llamada wavelet, donde (Ec 2.10)

$$\psi_{jk}(t) = 2^{j/2}\psi_{00}(2^j t - k)$$

Debido a que $V_{j+1} = V_j \oplus W_j$, entonces ψ puede ser escrito en términos de $\phi(2t - k)$, el cual forma una base para $V_{(j-1)}$. La siguiente relación para ψ es análoga a la relación dos-escala para ϕ:

$$\psi(t) = \sum_k h_1(k)\sqrt{2}\varphi(2t - k) \qquad (2.14)$$

Esta es la relación dos-escala para wavelets.

Ejemplo 2.10. Dadas dos funciones $\phi(2t)$y $\phi(2t-1)$en Fig. 2.27, y dados los filtros h_0 y h_1 como:

$$h_0(k) = \left\{\frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{2}}\right\}$$
$$h_1(k) = \left\{\frac{1}{\sqrt{2}} \quad -\frac{1}{\sqrt{2}}\right\}$$

Use las ecuaciones 2.13 y 2.14 para encontrar la función de escalamiento $\phi(t)$ y la "wavelet madre" $\psi(t)$

$$= \sum_k h_0(k)\sqrt{2}\,\varphi(2t-k)$$
$$= \underbrace{\tfrac{1}{\sqrt{2}}\sqrt{2}\,\varphi(2t)}_{k=0} + \underbrace{\tfrac{1}{\sqrt{2}}\sqrt{2}\,\varphi(2t-1)}_{k=1} = \varphi(2t) + \varphi(2t-1)$$

Solución:

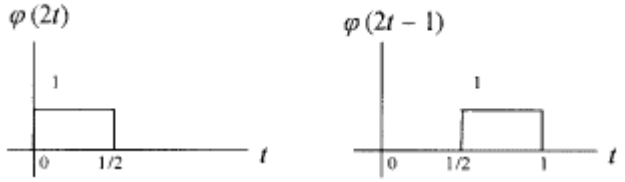


Figura 2.27

Esto resulta en $\phi(t)$, mostrado en la Fig 28a. Además,

$$\psi(t) = \sum_k h_1(k)\sqrt{2}\,\varphi(2t-k)$$
$$= \underbrace{\tfrac{1}{\sqrt{2}}\sqrt{2}\,\varphi(2t)}_{k=0} - \underbrace{\tfrac{1}{\sqrt{2}}\sqrt{2}\,\varphi(2t-1)}_{k=1} = \varphi(2t) - \varphi(2t-1)$$

Esto resulta en $\psi(t)$, mostrado en la Fig. 2.28b.

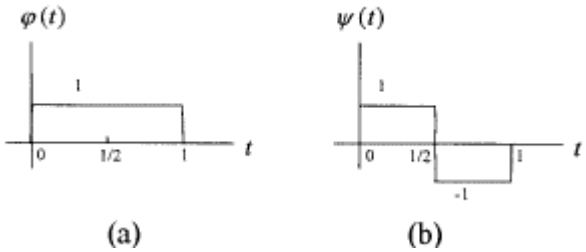


Figura 2.28

Notar que las ecuaciones 2.13 y 2.14 se mantienen a cualquier escala. En un número dado de coeficientes $h_0(k)$y $h_1(k)$ relacionan las funciones de escalamiento y wavelets a una resolución a las funciones de escalamiento a la siguiente resolución más baja.

La transformada Wavelet Continua está definida como la suma a través de todo el tiempo de la señal multiplicado por las versiones escaladas y recorridas de la función wavelet ψ [Misiti et al. 2010].

$$\mathcal{W}_\psi[s](a,b) = \frac{1}{\sqrt{a}} \int_{\mathbb{R}} s(t)\overline{\psi\left(\frac{t-b}{a}\right)}\, dt,$$

Donde *s* es la señal, *a* es la escala y *b* la rotación [Jansen-2005].

El resultado de la transformada wavelet continua son muchos coeficientes wavelet *w* (Figura 2.29), los cuales son una función de escala y posición.

Multiplicando cada coeficiente por el apropiado wavelet escalado y recorrido, produce las constituyentes wavelets de la señal original.

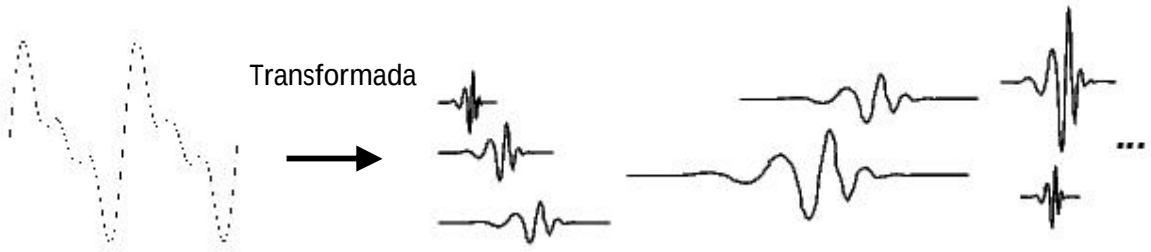


Figura 2.29 - Descomposición de la señal es sus wavelets constituyentes

2.1.3.1 Escalamiento

Escalar una wavelet implica estirarla o comprimirla, se introduce un *factor de escala*, denotado usualmente por la letra *a* (Figura 2.30).

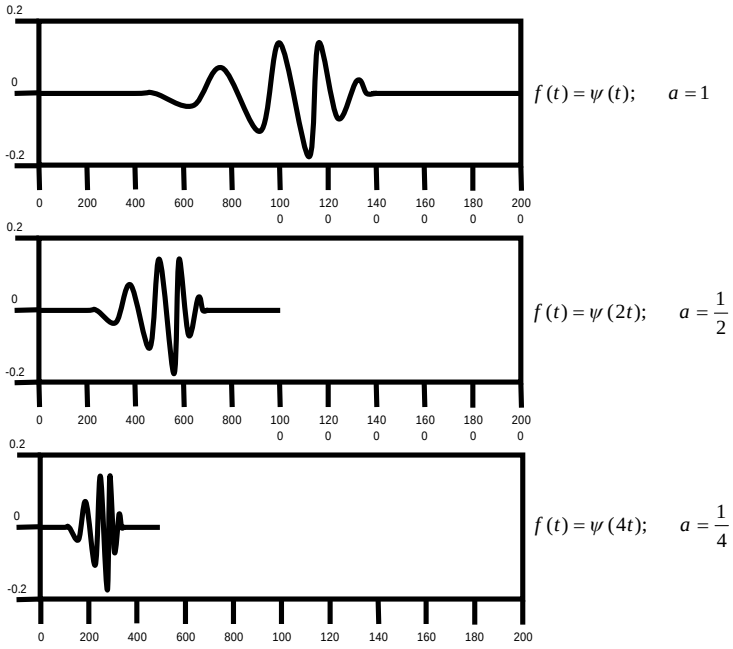


Figura 2.30 - Función wavelet escalada con un factor *a* de 1, ½, ¼

Este factor de escala funciona de manera igual con los wavelets. Entre más chico el factor de escala, la wavelet será más comprimida y viceversa (Misiti, 2010).

2.1.3.2 Retardo

Retardar una wavelet significa retrasar o adelantar su aparición. Matemáticamente, retrasar una función *f*(*t*) por k es representado por *f*(*t*-*k*) (Figura 2.31).

Figura 2.31 - Función wavelet original y función wavelet retardada

2.1.3.3 Aplicaciones Wavelet

Las wavelets tienen aspectos de escala y tiempo, es por eso que cada aplicación de ellas tiene aspectos de este tipo.

a) Aspectos de escala

Como un complemento del análisis espectral de la señal, nuevas formas de señales aparecen. Estas son señales menos regulares que las usuales.

La señal cúspide presenta una variación local muy rápida. Su ecuación es t^r con t cercado a 0 y $0 < r < 1$. Entre más bajo sea el valor de r, más definida será la señal.

Algunos dominios que usan las técnicas wavelets para el estudio de la regularidad son:

- Biología, para el estudio de la membrana celular con el fin de distinguir la normal de la patógena.
- Metalurgia, para la caracterización de superficies rugosas.
- Finanzas (lo cual es sorprendente), para la detección de propiedades de la variación rápida de valores.
- En la descripción del trafico de Internet, para diseñar el tamaño de los servicios.

b) Aspectos de tiempo
Las metas principales son:

- Detección de bordes y rupturas.
- Estudios de fenómenos de corto-tiempo como los procesos transitorios.

Como aplicaciones de dominio, se tiene:

- Reconocimiento automático de blancos.
- Detección de eventos patológicos cortos como crisis epilépticas.
- Revisar el ruido excesivo en ruedas dentadas, y más generalmente en procesos de control de calidad.

2.1.3.4 Tipos de Wavelets

a) Wavelets Grossman-Morlet de tiempo-escala

El análisis de tiempo-escala involucra el uso de un vasto rango de escalas. Esta noción de escala, implica que la señal es reemplazada, en una escala dada, por la mejor aproximación posible que pueda ser dibujada a esa escala. Al pasar de las escalas largas hacia las escalas finas, uno hace un acercamiento y se llega a representaciones más y más precisas de la señal dada.

Estas wavelets se definen **[Jaffard et al.]** al empezar con una función ψ de la variable real *t.* Esta función es llamada la wavelet madre si está bien localizada y oscilante. La condición de localización esta expresada en la manera usual al decir que la función decrementa rápidamente a cero cuando $|t|$ tiende a infinito. La segunda condición sugiere que ψ vibra como una onda. Matemáticamente, se requiere que la integral de ψ sea cero y que los otros primeros m momentos de ψ también desaparezcan. Es expresado por las relaciones

$$\int_{-\infty}^{\infty} t^n \psi(t) dt = 0 \quad \text{para} \quad n = 0,1,...,m\text{-}1$$

La wavelet madre ψ genera los otros wavelets de la familia $\psi_{(a,b)}, a>0, b\in\mathbb{R}$, por cambio de escala y traslación en tiempo. Entonces se tiene

$$\psi_{(a,b)}(t)=\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \qquad a>0, \qquad b\in\mathbb{R}.$$

Alex Grossmann y Jean Morlet que cualquier señal de energía finita puede ser representada como una combinación lineal de wavelets $\psi_{(a,b)}$ y que los coeficientes de esta combinación son, los productos escalares de $\int_{-\infty}^{\infty} f(t)\overline{\psi}_{(a,b)}(t)dt.$ Estos productos escalares miden, en algún sentido, la fluctuación de la señal f alrededor del punto *b*, en la escala dada por $a > 0$.

Los siguientes ejemplos de diferentes tipos de wavelets han sido tomados Wavelet Toolbox™ 4 User's Guide de [Misiti et al. 2010]

b) Haar

La wavelet Haar es la más simple de todas, no es continua, y se asemeja a una función paso. (Figura 2.32)

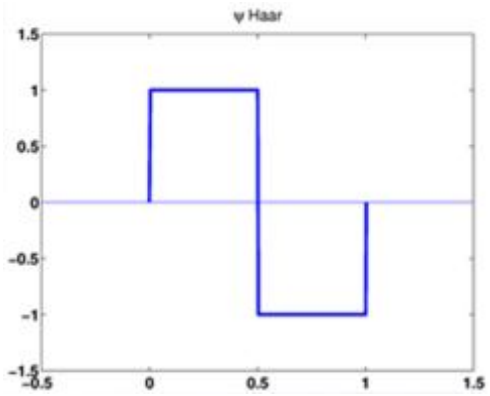


Figura 2.32 - Wavelet Haar

c) Daubechies

Una de las mentes más brillantes en el campo de la investigación wavelet, es Ingrid Daubechies, quien invento las llamadas wavelets ortonormales de soporte compacto.

Los nombres de la familia de Wavelets de la familia de Daubechies son escritas de la forma dbN, donde N es el orden. La wavelet db1, es la misma que la wavelet Haar. A continuación se muestran algunas wavelet dbN, con N > 1 (Figura 2.33).

Db4 **Db20**

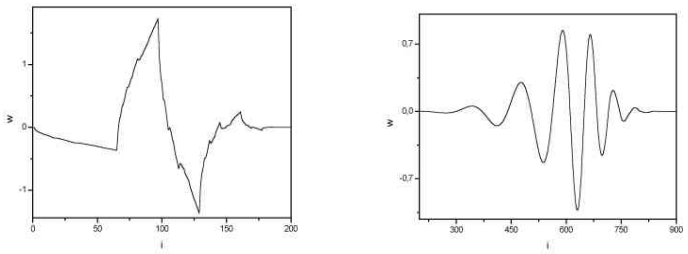


Figura 2.33 - Wavelet Dbn, para n = 4 y n= 20.

c) Coiflets

Esta función Wavelet tiene 2N momentos iguales a 0 y la función de escalamiento tiene 2N-1 momentos iguales a 0 (Figura 2.34). Las dos funciones tienen soporte de tamaño 6N-1.

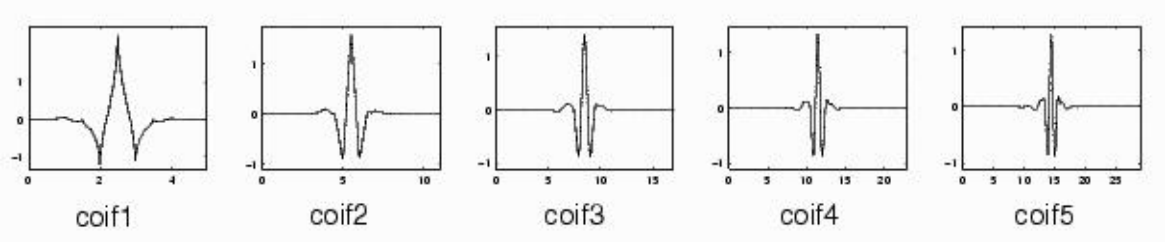


Figura 2.34 - Wavelet Coiflet

d) Symlets

Las symlets son wavelets casi simétricas, propuestas por Daubechies como una modificación a la familia db. He de ahí que estas dos familias de wavelets tengan propiedades muy similares (Figura 2.35).

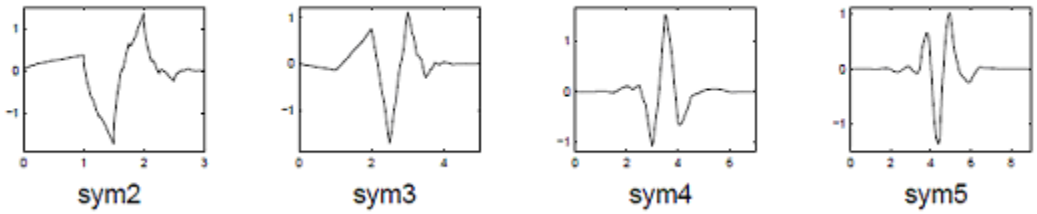


Figura 2.35 - Wavelet Symlets

e) Mexican Hat

El nombre de esta wavelet es debido a que se asemeja a un sombrero mexicano (Figura 36), esta wavelet no tiene una función de escalamiento y es derivada de una función que es proporcional a la segunda función derivativa de la función gaussiana de probabilidad de densidad.

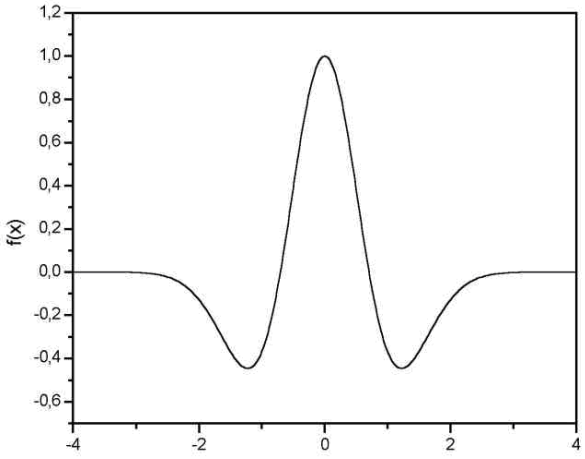


Figura 2.36 - Wavelet Mexican Hat

2.1.4 El porqué de los wavelets para el procesamiento de señales de habla

Con base en el analisis multiresolución, una señal es descomprimida en una aproximación y detalles a varias escalas, varios niveles de información a traves de resoluciones sucesivas se pueden extraer al descomponer la señal original usando una wavelet con base ortonormal. Como una transformada convencional, la transformada de Fourier de corto tiempo es usada ampliamente en las matemáticas e ingeniería; una de sus limitantes es que usa una sola ventana para todas las frecuencias, la resolucion para este análisis es la misma para todos los lugares en el plano de la frecuencia. Esta limitante represanta una desventaja en las señales de habla y audio, dado que el sistema del oido humana usa una resolucion que depende de la frecuencia. La transformada Wavelet discreta puede resolver este incoveniente con el analisis multiresolución. [Pham, 2007]

El análisis flexible en el plano de tiempo-frecuencia de la Transformada discreta Wavelet en comparación con la transformada de Fourier de corto tiempo muestra sus ventajas para el procesamiento de habla. Dado que las funciones base wavelet son ondas cortas y generadas al escalar la wavelet madre, son muy bien localizadas en los dominios del timpo y escala. Este comportamiento automatico de la descomposición wavelet es

apropiado para el procesamiento de señales de habla que requieren una resolucion de alta frecuencia para analizar los componentes de baja frecuencia(sonidos de habla), y altas resoluciones temporales para analizar componentes de alta frecuencia. Este comportamiento inteligente de la Transformada discreta Wavelet es empleada para representaciones auditivas de señales acústicas, codificación de habla usando representacion de paquetes wavelet en el contexto de modelado auditivo, segmentacion de habla y clasificación fonética. [Pham, 2007]

2.1.5 Señales de potencia y energía

Los siguientes temas han sido extraídos del libtro *Elements of Wavelet for Engineers and Scientists* **[Mix and Olejniczak, 2003]** donde se relacionan los diferentes tipos de señales con su respectiva transformada.

Una resistencia disipa energía eléctrica como calor, pero también la energía es disipada sobre el tiempo. Si el voltaje no es muy grande, la resistencia se templa a una temperatura constante y se mantiene. En otras palabras, llega a un estado fijo. Así, de este modo, la *energía promedio* es disipada en la resistencia.

Para una señal de tiempo continua voltaje $v(t)$ impresionado a través de una resistencia, el poder promedio es definido por

$$P = \lim_{a\to\infty} \frac{1}{2a} \int_{-a}^{a} \frac{v^2(t)}{R} dt \qquad (2.15)$$

Para que esta cantidad dependa solamente de la señal, se coloca la resistencia con valor igual a 1 ohm. La potencia total, potencia promedio, o el valor cuadrático medio de la señal se define por

$$P = \lim_{a \to \infty} \frac{1}{2a} \int_{-a}^{a} v^2(t) dt \qquad (2.16)$$

La raíz cuadrada de *P* es el valor de la raíz cuadrada media (rms). Note que la corriente a través de la resistencia puede ser usada tan fácilmente como el voltaje en la definición de potencia. La definición de potencia en una señal se aplica a cualquier señal, sea de voltaje, corriente, o cualquier otra función no eléctrica tal como presión o velocidad.

Esto sirve para introducir la clasificación de las señales en una de dos categorías: señales de potencia o señales de energía. Una señal $v(t)$ es llamada señal de potencia si la expresión en la Ec. 2.16 es finita. Una señal $v(t)$ es llamada señal de energía si la energía es finita, donde la energía E está dada por

$$E = \int_{-\infty}^{\infty} v^2(t) dt \qquad (2.17)$$

Esta es la energía total, no la energía instantánea. Cualquier señal que dure solo una duración finita es una señal de energía. Una señal, que tiene una duración infinita puede ser una señal de potencia o una señal de energía, dependiendo del valor de las expresiones en Ecs. 2.16 y 2.17

La potencia en una señal periódica de periodo T es dada por la fórmula simplificada

$$P = \frac{1}{T} \int_{0}^{T} v^2(t) dt \qquad (2.18)$$

Para señales de tiempo discretas, estas definiciones se convierten en

$$P = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} v^2(n) \qquad (2.19)$$

y

$$E = \sum_{n=-\infty}^{\infty} v^2(n) \qquad (2.20)$$

En la Ec. 2.19, la suma sobre el rango (-*N*, *N*) tiene 2*N* + 1 términos, se divide entre este numero de términos para encontrar el promedio. Esto también es llamado el valor medio-

cuadrado. El valor medio cuadrado es la potencia promedio, aunque el concepto de potencia pierde su significado para señales de tiempo discretas. La energía en Ec. 2.20 es la energía *total*, no la energía instantánea.

La potencia en una señal periódica con periodo N puede ser encontrada al promediar sobre un solo periodo.

$$P = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n) \qquad (2.21)$$

2.1.5.1 Relación entre Transformadas de Fourier y Haar.

Las señales caen naturalmente en una de cuatro categorías:

1. Señales de energía de tiempo continuas.
2. Señales de potencia de tiempo continuas.
3. Señales de energía de tiempo discretas.
4. Señales de potencia de tiempo discretas.

Existe una forma de la transformada de Fourier para cada categoría.

- Transformada de Fourier de tiempo continua (CTFT).
- Series de Fourier de tiempo continua (CTFS).
- Transformada de Fourier de tiempo discreta (DTFT).
- Series de Fourier de tiempo discreta (DTFS),

La relación es la siguiente:

1. Señales de energía de tiempo continuas $\leftrightarrow$CTFT.
2. Señales de potencia de tiempo continuas $\leftrightarrow$CTFS
3. Señales de energía de tiempo discretas $\leftrightarrow$DTFT.
4. Señales de potencia de tiempo discretas $\leftrightarrow$DTFS.

La transformada rápida de Fourier es una versión rápida de la DTFS y no se aplica a ninguna de las otras transformadas. La DTFS está definida por

$$V(k) = \sum_{n=0}^{N-1} v(n) e^{-j2\pi nk/N} \qquad (2.22)$$

La ecuación 2.22 es realmente N ecuaciones, una para cada valor de *k*.

$$V(0) = \sum_{n=0}^{N-1} v(n) e^{0} = \sum_{n=0}^{N-1} v(n)$$

$$\text{v(1)} = \sum_{n=0}^{N-1} v(n) e^{-j2\pi n/N}$$

$$\vdots \qquad\qquad \vdots$$

$$\text{v(N-1)} = \sum_{n=0}^{N-1} v(n) e^{-j2\pi n(N-1)/N}$$

Hay un término exponencial diferente usado en cada ecuación, empezando con e^{0}, $e^{-j2\pi n/N}$, y progresando hasta $e^{-j2\pi n(N-1)/N}$. Estas son las funciones base para la expansión. La DTFS es una expansión de v(n) en términos de estos exponenciales. Sea

$$W_N = e^{-j2\pi/N} \qquad (2.23)$$

Entonces

$$W_N^0 = e^0$$
$$W_N^1 = e^{-j2\pi/N}$$
$$W_N^2 = e^{-j2\pi 2/N}$$
$$\vdots \qquad\qquad \vdots$$
$$W_N^{N-1} = e^{-j2\pi(N-1)/N}$$

Estas N ecuaciones puede ser puestas en forma de matriz. Por ejemplo, si N=4, la matriz está dada por

$$\begin{bmatrix} V(0) \\ V(1) \\ V(2) \\ V(3) \end{bmatrix} = \begin{bmatrix} w_4^0 & w_4^0 & w_4^0 & w_4^0 \\ w_4^0 & w_4^1 & w_4^2 & w_4^3 \\ w_4^0 & w_4^2 & w_4^4 & w_4^6 \\ w_4^0 & w_4^3 & w_4^6 & w_4^9 \end{bmatrix} \begin{bmatrix} V(0) \\ V(1) \\ V(2) \\ V(3) \end{bmatrix} \qquad (2.24)$$

Usando esta notación, la Ec. 2.22 puede ser escrita como

$$V(k) = \sum_{n=0}^{N-1} v(n) W_N^n \qquad (2.25)$$

2.1.6 Transformada Haar.

La transformada Haar es una transformada wavelet que puede ser calculada con una fórmula. La cual está dada por:

$$c_{00} = \int_0^1 v(t)\varphi_{00}(t)dt \qquad (2.26a)$$

$$d_{kj} = \int_0^1 v(t)\psi_{kj}(t)dt \qquad (2.26b)$$

Donde las funciones base φ_{00} y ψ_{kj}, son mostradas en la Fig. 2.37. La función $\varphi_{00}(t)$ es llamada la función de escalamiento, y la función $\psi_{kj}(t)$ es llamada wavelets.

Este diagrama muestra el nivel 0, 1 y 2 de las funciones base Haar, pero estas pueden continuar a niveles más y más altos. El nivel 0 contiene las funciones $\varphi_{00}(t)$ y $\psi_{kj}(t)$. El nivel 1 contiene dos funciones $\psi_{10}(t)$ y $\psi_{11}(t)$. El nivel 2 contiene las cuatro funciones $\psi_{20}(t)$, $\psi_{21}(t)$, $\psi_{22}(t)$ y $\psi_{23}(t)$. Cada nivel más alto contiene lo doble de wavelets $\{\psi_{jk}\}$. Entonces, el nivel 3 contiene ocho wavelets, cada uno la mitad de largo que los wavelets de nivel 2 y escalado apropiadamente para tener unidad energía. Para calcular la transformada Haar, expandimos la función tiempo en términos de las funciones base. La ecuación 2.26 da las formulas para calcular la transformada.

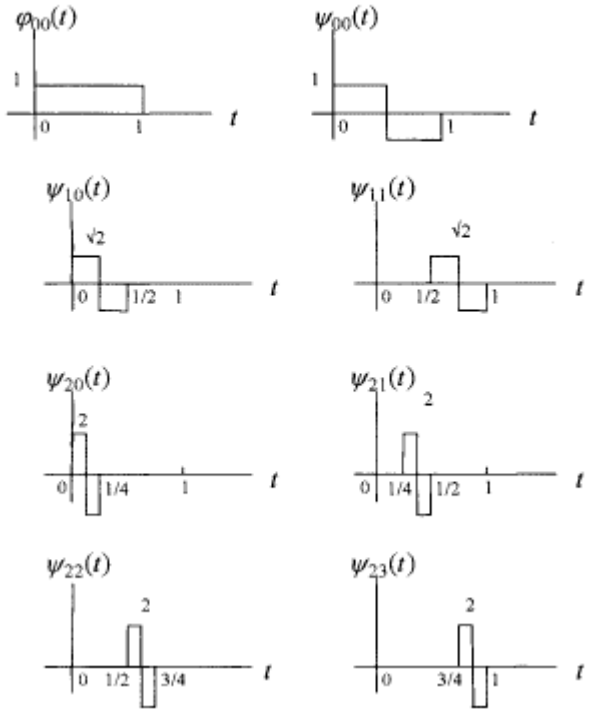


Figura 2.37 - Funciones base Haar para nivel 2

2.2 Minería de datos

Los temas presentados sobre minería de datos, fueron basados en el document A Survey on Wavelet Applications in Data Mining [Li-2002], el cual es un compendio de las técnicas mas usadas para tratar este problema.

El proceso de minería de datos, es la extracción de patrones útiles de una gran cantidad de información. Puede ser vista como una actividad multidisciplinaria porque hace uso de diferentes disciplinas de la inteligencia artificial, como aprendizaje automático, sistemas expertos, reconocimiento de patrones, también usa disciplinas matemáticas, como la estadística. Los wavelets se pueden usar el proceso de minería de datos, por su propiedad particular de tener una estructura de descomposición jerárquica y de multiresolución. Puede proveer soluciones considerablemente más eficientes a muchos

problemas de la minería de datos. Las wavelet pueden proveer representaciones de datos que pueden hacer el proceso más eficiente y preciso.

2.2.1 Manejo de datos.

La diferencia entre la minería de datos con otros tipos de análisis de datos, es la inmensa cantidad de datos. Debido a esto, un aspecto muy importante de la minería de datos, es el manejo de los datos. El propósito de un buen manejo de datos es encontrar métodos que faciliten el acceso eficiente y rápido. La propiedad de la transformación wavelet puede aplicarse al manejo de los datos, ya que proporciona una estructura jerárquica natural y una representación multidimensional de los datos.

2.2.2 Preprocesamiento

Por lo regular, en los problemas del mundo real, no se pueden aplicar algoritmos de minería de datos directamente al conjunto de datos, estos contienen ruido, valores vacíos o inconsistentes, además, que los datos del mundo real tienden a ser demasiado grandes, de alta dimensionalidad, es por eso que se necesita hacer un preprocesamiento de los datos.

Con el preprocesamiento podemos limpiar el conjunto de datos, para remover el ruido, hacer una reducción de datos para reducir la alta dimensionalidad de los datos, incluso podemos transformar el conjunto de datos, a una representación más adecuada para el proceso de minería de datos. Con la propiedad que tienen los wavelets de tener momentos de desvanecimiento (**vanishing points**), conocemos de antemano que en la mayoría de los casos solo algunos coeficientes wavelets son importantes. Al solo tener los coeficientes wavelets significativos, la transformada wavelet se puede aplicar para quitar el ruido y ayudar en la reducción de la dimensionalidad. Debido a la otra propiedad de los wavelets, de tener sus coeficientes no correlacionados, el conjunto de datos original se

puede transformar al dominio wavelet y entonces hacer tareas de minería de datos en este nuevo dominio.

2.2.3 Remover ruido

Los datos ruidosos pueden aparecer de muchas maneras, como errores en la medición durante la adquisición de los datos, errores humanos o de computadora al momento de registrar datos, fenómenos de la naturaleza, etc. Quitar el ruido de los datos, es un proceso donde primero se tiene que identificar los datos ruidosos, datos extraños, para posteriormente hacer un proceso que elimine estas alteraciones en los datos. Existen varias técnicas para lidiar con el ruido, agrupamiento, detección de outliers, etc. Hay técnicas para detectar datos outliers, como es el agrupamiento, y los datos que no entran en ningún grupo se toman como outliers. Las tecnicas que usan wavelets proveen una manera muy efectiva de quitar el ruido, la idea principal detrás de quitar el ruido mediante wavelets, es transformar los datos en una base diferente, la base wavelet, donde se sabe que grandes coeficientes son principalmente la información útil, y los pequeños representan ruido. Entonces al modificar estos coeficientes en la nueva base, se puede trabajar directamente con el ruido que tiene el conjunto de datos [Li-2002].

2.2.4 Transformación de los datos

El transformar los datos al dominio wavelet tiene muchas ventajas, se tiene una amplia gama de operaciones disponibles para este dominio, operando en los coeficientes de la transformada wavelet del conjunto de datos original. Con la propiedad de multiresolución de los wavelet, se puede operar en diferentes resoluciones, manipular características de los datos en diferentes escalas, etc. Realizar las operaciones en este dominio, es mucho más sencillo que realizar la misma operación en los datos originales. En conclusión, es mucho más cómodo, y se tiene mucha ventaja el trabajar en el dominio wavelet , que en el dominio original de los datos.

2.2.5 Reducción de la dimensionalidad

En términos generales reducir la dimensionalidad consta de tomar un conjunto de datos inmenso, para representarlo usando un conjunto de datos más pequeño con muy poca o nula pérdida de información. Se ha dicho que a partir de la transformada wavelet, se pueden retener solamente los coeficientes más significativos, entonces hay una reducción de la dimensionalidad, al descartar los coeficientes que tienen muy poca información.

Existen dos maneras para la reducción de la dimensionalidad usando wavelets: [Li 2002]

- Mantener los *k* coeficientes más grandes y aproximar el resto con 0,
- Mantener los primeros k coeficientes y aproximar el resto con 0.

2.2.6 Tareas de minería de datos y algoritmos.

Son los procedimientos que se aplican al conjunto de datos para extraer información útil. Existen diferentes tareas relacionadas con la minería de datos. Como es sabido con los algoritmos, no existen un algoritmo universal que funcione para todo tipo de problemas, es por eso que existen diferentes algoritmos que sirven para una sola tarea.

a) Agrupamiento

Gracias a la propiedad de multiresolución que poseen los wavelets, se han creado algoritmos que identifican grupos a diferentes escalas. Desde el punto de vista del procesamiento de señales, si la colección de objetos en el espacio de características es visto como una señal *n* dimensional, las partes de la señal con altas frecuencias corresponde a las regiones del espacio de características donde hay un rápido cambio en la distribución de los objetos (los limites de los grupos), y las partes de baja frecuencia de la señal que tienen alta

amplitud, corresponden a las áreas del espacio de características donde los objetos están concentrados (los grupos).

b) Clasificación

El objetivo de la clasificación es encontrar características de los grupos, a la que cada instancia pueda pertenecer, de esta forma comprendiendo los datos existentes y predecir el grupo al cual las nuevas instancias pertenecerán. Los wavelets pueden ser usados para las tareas de clasificación, al transformar el conjunto de datos originales al dominio wavelet para aplicar los métodos de clasificación. La propiedad de multiresolución se puede incorporar en los métodos de clasificación para ayudar al proceso.

c) Visualización

La visualización es un proceso de la minería de datos que permite el entender mejor el conjunto de datos al visualizar por medio de graficas el comportamiento en lugar de texto o números. Sin embargo, no siempre es posible lograr una visualización simple de los datos, debido a la gran dimensionalidad que pueden tener. La transformación multiescala wavelet facilita el acceso al conjunto de datos, con una visualización más simple, al ver las características más significativas primero.

2.2.7 WEKA

Los conceptos que se tratan respecto a WEKA fueron tomados de referencia del libro "Data Mining: Practical machine learning tools and techniques" **[Witten y Frank 2005],** el cual tiene capítulos dedicados enteramente al funcionamiento de este software.

2.2.7.1 Algoritmos de aprendizaje

WEKA utiliza diferentes tipos de clasificadores, divididos en diferentes secciones, como son: Clasificadores bayesianos, árboles, reglas, que se pueden observar en la Tabla 2.1, Tabla 2.2 y Tabla 2.3 respectivamente. Se han hecho experimentaciones con NaiveBayes, siguiendo la línea trazada por los trabajos que han experimentado con la transformada Wavelet para abordar este problema, pero es importante conocer los posibles clasificadores para en un futuro hacer experimentaciones modificando el clasificador.

Tabla 2.1 - Clasificadores bayesianos soportados por WEKA.

Nombre	Función
AODE	Estimadores promedio, de una dependencia
BayesNet	Aprender redes Bayesianas
ComplementNaiveBayes	Construir un clasificador complementario Naive Bayes
NaiveBayes	Clasificador de probabilidad estándar Naive Bayes
NaiveBayesMultinomial	Versión multinomial de Naive Bayes
NaiveBayesSimple	Implementación simple de Naive Bayes
NaiveBayesUpdateable	Clasificador Naive Bayes incremental que aprende una instancia a la vez

Tabla 2.2 - Clasificadores de árbol soportados por WEKA

Nombre	Función
ADTree	Construye árboles de decisión alternantes
DecisionStump	Construye árboles de decisión de un nivel
Id3	Algoritmo de arbole de decisión básico de divide y vencerás
J48	Aprendizaje de Árbol de decisión C4.5
LMT	Construye árboles de modelos logísticos
M5P	Aprendizaje de árbol modelo M5
Nbtree	Construye un árbol de decisión con clasificadores Naive Bayes en las hojas
RandomForest	Construye bosques aleatorios
RandomTree	Construye un árbol que considera un numero dado de características aleatorias en cada nodo
REPTree	Aprendizaje de árbol rápido que usa acotamiento de error reducido
UserClassifier	Permite a los usuarios construir su propio árbol de decisión.

Tabla 2.3 - Clasificadores basados en reglas soportados por WEKA.

Nombre	Función
ConjunctiveRule	Aprendizaje de regla simple conjuntiva
DecisionTable	Construye un clasificador de tabla de decisión simple
Jrip	Algoritmo RIPPER para una rápida y efectiva inducción de regla
M5Rules	Obtiene reglas a partir de moles de árboles construidos usando M5
Ninge	Método del vecino más cercano para generar reglas
OneR	Clasificador 1R
Part	Obtiene reglas a partir de árboles de decisión parciales usando J4.8
Prism	Algoritmo simple de cubrimiento para reglas
ZeroR	Predice la clase mayoritaria (si es nominal) o el valor promedio (si es numérico)

2.2.7.2 La estructura de WEKA

Para adentrarse más a WEKA y poder manejarlo de una mejor forma, es necesario el aprender algo sobre cómo está compuesto el software.

El paquete *weka.core* es central para el sistema WEKA y sus clases son accedidas desde casi cualquier otra clase. Las clases clave en el paquete *core* son *Attribute*, *Instance* e *Instances*. Un objeto de la clase *Attribute* representa un atributo. Contiene el nombre del atributo, su tipo y sus posibles valores en el caso de ser un valor nominal o atributo cadena. Un objeto de la clase *Instance* contiene los valores de atributo de una determinada Instancia; y un objeto de la clase *Instances* almacena un conjunto ordenado de instancias, en otras palabras, el conjunto de datos.

El paquete weka.classifiers contiene implementaciones de la mayoría de los algoritmos para clasificación y predicción numérica descrita en el libro. La clase más importante en este paquete es *Classifier*, que define la estructura general de cada esquema para clasificación o predicción numérica. Classifier contiene tres métodos, *buildClassifier()*, *classifyInstance()*, y *distributionForInstance()*. Cada esquema redefine estos métodos de acuerdo a como construye su clasificador y como clasifica las instancias.

Existen otros paquetes que vale la pena mencionar: *weka.associations*, *weka.clusterers*, *weka.estimators*, *weka.filters*, y *weka.attributeSelection*. El paquete weka.associations contiene asociaciones de reglas de aprendizaje. Estas han sido acomodadas en un paquete diferente porque las reglas de asociación son fundamentalmente diferentes para cada clasificador. El paquete *weka.clusterers* contiene métodos para aprendizaje no supervisado. El paquete *weka.estimators* contiene subclases de una clase genérica *Estimator*, la cual computa diferentes tipos de distribución de probabilidad. Estas subclases son usadas por el algoritmo Naive Bayes (entre otros).

Existen diferentes opciones en la línea de comandos que son importantes conocer, existen dos tipos de opciones, las opciones genéricas que se pueden usar con cualquier esquema de aprendizaje, y las opciones específicas, que se aplican únicamente a esquemas particulares. Si invocamos un esquema sin especificar alguna opción, se despliega en pantalla todas las opciones aplicables: primero las opciones generales, después las opciones especificas.

2.3 Medidas estadísticas: curtosis y *skewness*.

Una tarea fundamental en varios análisis estadísticos es caracterizar la localización y variabilidad de un conjunto de datos. Una caracterización adicional de los datos incluye la curtosis y *skewness*.

La skewness es una medida de simetría, o más precisamente, de falta de simetría. Una distribución o un conjunto de datos, es simétrico si este se mira de la misma manera de izquierda a derecha desde un punto (normalmente céntrico).

La curtosis es una medida de si los datos son óptimos ó planos relativos a la distribución normal. Un conjunto de datos con curtosis alta tienden a tener picos distintivos cerca de la media, declinar bastante rápido, tener colas pesadas. Los conjuntos de datos con curtosis baja tienden a tener un plano alto cerca de la media más que un pico puntiagudo. Una distribución uniforme sería un caso extremo.

Definición de skewness. Para datos univariados $Y_1, Y_2,\ldots,Y_N$, la fórmula de la skewness es:

$$Skewness\ c_i = \frac{\sum_{k=1}^{S}(c_{ik} - Pc_i)^3}{(S-1)(Dc_i)^3}$$

donde Pc_i es la media, *Dci* es la desviación estándar y *S* es el número de datos. La skewness de una distribución normal es cero, y cualquier dato simétrico debe tener una skewness cerca de cero. Los valores negativos en la skewness indican que los valores están sesgados hacia la izquierda y los valores positivos indican que los valores están sesgados a la derecha. Por *sesgados a la izquierda*, se entiende que la cola izquierda es larga en comparación con la cola derecha. Algunas medidas tienen un límite inferior y están sesgadas a la derecha. Por ejemplo, en estudios de confiabilidad/fiabilidad, los tiempos de fracaso no pueden ser negativos.

Definición de curtosis. Para datos univariados $Y_1, Y_2,\ldots,Y_N$, la fórmula de la curtosis es:

$$Kurtosis\ c_i = \frac{\sum_{k=1}^{S}(c_{ik} - Pc_i)^4}{(S-1)(Dc_i)^4}$$

donde Pc_i es la media, Dc_i es la desviación estándar y *S* es el número de datos.

Capítulo 3 Estado del Arte

En este capítulo se hablará sobre los diferentes acercamientos que se han usado para resolver el problema de LID: Prosódico, acústico, fonotáctico y LVCSR. En el trabajo de Reyes [Reyes-Herrera 2007] así como en el trabajo de Vargas [Vargas-Martínez] se realiza una perspectiva histórica de las personas que han trabajado en LID y sus metodologías, desde sus comienzos en la década de 1970, hasta el uso de la Transformada Wavelet en este campo.

Los diferentes acercamientos usados para resolver LID están basados en el libro, "Multi-lingual Speech Processing", [Schultz y Kirchoff 2006] que trata específicamente de procesamiento de habla multilingüe, con un capítulo en específico enfocado a el problema de LID, donde se hace un estudio del estado del arte de este problema,

3.1 Diferentes acercamientos usados para resolver LID

El problema principal en resolver la tarea de LID es reducir la complejidad del lenguaje humano de tal forma que un algoritmo pueda identificarlo a partir de una muestra de audio corta [Navrátil]. Existen diferencias entre los lenguajes, que varían desde las más fáciles de identificar como el uso de palabras totalmente diferentes, hasta variaciones más sutiles.

El modelo ideal para un sistema automático LID es el es oyente humano. Los humanos son capaces de reconocer un lenguaje aun de muestras muy cortas, suponiendo que tienen un cierto grado de familiaridad con el lenguaje, aun con muy poca exposición a la lengua, los oyentes pueden identificar el lenguaje en cuestión.

3.1.1 Bases de datos para la evaluación.

La disponibilidad de un gran corpus de habla es uno de los mayores factores en la investigación del habla. El tener una base de datos estandarizada no es un proceso trivial, se

necesita una metodología para recolectar las muestras de voz, y clasificarlas, en la base del trabajo por [Yeshwant Muthusamy et al. 1992], se creó la primer base de datos multilingüe para la investigación LID en el centro para el entendimiento de legua hablada en el Instituto de graduados de Oregon (OGI) y se liberó a la comunidad investigadora en 1993. Esta base de datos, una colección de habla telefónica en 11 lenguajes diferentes, proporcionó una manera unificada de comparar y evaluar algoritmos LID y reemplazó la entonces común práctica de reportar resultados aislados en diferentes conjuntos de datos propietarios. Como resultado, el volumen de publicaciones técnicas en LID incremento notablemente después de 1993, lo cual demostró la significancia del "factor datos" [Muthusamy et al., 1994]. La salida de la primera base de datos OGI fue seguida por una serie de colecciones de datos subsecuentes.

En 1996-1997, el consorcio lingüístico de datos (LDC) organizo una serie de colecciones de datos relevantes para LID de conversaciones de teléfono improvisado entre amigos en 6 (CallHome) y 12 (CallFriend) lenguajes. Hay cerca de 60 conversaciones por lenguaje, cada una durando entre 5 y 30 minutos. La base de datos CallFriend fue usada por el instituto nacional de estándar y tecnología (NIST) como una fuente de datos para conducir una evaluación de algoritmos LID, organizado en 1996 y 2003.

3.1.2 Acercamientos acústicos

Los LID puramente acústicos se enfocan en capturar las diferencias esenciales entre los lenguajes al modelar directamente distribuciones de características espectrales. Esto se logra al extraer un conjunto de características espectrales independientes del lenguaje a partir de segmentos de habla y usando un clasificador estadístico para identificar los patrones dependientes del lenguaje en tales características.

3.1.2.1 Modelado acústico con modelos de mezclas gaussianas

Los modelos de mezclas gaussianas (GMM) se han usado extensivamente en reconocimiento de habla, aunque su implementación varia, por ejemplo, dependiendo en el

tipo de información fonética modelada o la inclusión de estructuras temporales mediante modelos ocultos de Markov (HMM). Debido al hecho de que los GMMs pueden aproximar cualquier función de densidad arbitraria y que existen poderosos algoritmos de entrenamiento para las estructuras GMM, estos modelos son la elección de preferencia en muchos sistemas LID del estado del arte. Algunos investigadores que han usado este enfoque son: Zissman (1993), Hazen y Zue (1997), Navrátil y Zühlke(1998) y más recientemente Torres Carrasquillo (2002).

3.1.2.2 Modelado acústico con modelos ANNs y SVMs

Existen alternativas en el modelado de la distribución de vectores de características acústicas dado el lenguaje, como las redes neuronales artificiales (ANNs) y máquinas de soporte vectorial (SVMs).

Las redes que se usan predominantemente en LID son las ANNs de perceptrón multicapa, aquellas en que la salida de una capa de la neurona se conecta a la siguiente capa solamente. ANNs pueden aproximar cualquier mapeo arbitrario, es por eso que presentan una herramienta poderosa de modelado [Funahasi, 1989].

Muthusamy [Muthusamy,1993] utilizó una ANN en la forma de perceptrón multicapas, la cual fue entrenada para mapear vectores de características de predicción perceptual lineal a una de siete amplias clases fonéticas, seguido de un alineamiento y segmentación. Estas clases amplias fueron usadas para realizar el LID.

En un estudio de 1998. Braun [Braun and Levkowitz, 1998] describió el uso de redes neuronales recurrentes (RNN) aplicados a segmentos perceptualmente importantes de muestras que fueron detectadas en una manera interactiva por los oyentes humanos. Literatura reciente sobre LID con ANNs acústicos es muy escasa, posiblemente debido al éxito de otros métodos y el alto número de muestras de entrenamiento requeridas por ANNs.

Las máquinas de soporte vectorial (SVMs) fueron introducidas por Vapnik [Vapnik, 1999] como un resultado analítico del problema empírico de minimización de riesgo. La esencia de los SVMs es la representación de los límites que separan las clases en un espacio dimensionalmente alto en términos de pocos puntos cruciales, obtenidos a partir de la muestra de entrenamiento, denominados vectores de soporte.

Campbell [Campbell et al, 2004] describió una aplicación de máquinas de soporte vectorial para la tarea de LID. El sistema SVM uso una expansión de kernel a partir de la entrada a un espacio de características de altas dimensiones, en la cual una clasificación lineal en vectores promediados es llevada a cabo. Los resultados mostraron un desempeño comparable o mejor a los GMMs y componentes fonotácticas [Campbell et al., 2004; Singer et al., 2003], por lo tanto representa un de las adiciones más prometedoras a las técnicas del estado del arte LID. En la tabla 3.1 se describen algunos sistemas que usan el enfoque acústico.

Tabla 3.1 - 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]

3.1.3 Modelado Fonotáctico.

Fonotácticas – las restricciones en frecuencias relativas de unidades de sonido y sus secuencias en el habla- es una de las fuentes de información LID ampliamente utilizadas. Su popularidad es dada a su implementación relativamente simple y escalable, y por su poder relativamente alto de discriminación de lenguajes. El método LID fonotáctico usa un marco de trabajo probabilístico y se basa en la propiedad inherente de cada lenguaje de exhibir fuertes frecuencias especificas del lenguaje y dependencias entre fonemas en un

sonido. Este fenómeno se puede ilustrar con un ejemplo: La combinación del fonema /i/ seguido de /ç/ ocurre frecuentemente en el alemán, y es inexistente en inglés. Relaciones similares entre unidades fundamentales de texto escrito (grafemas) fueron observadas por Markov en 1913 y las diferencias resultantes entre los lenguajes han sido analizadas en numerosos reportes lingüísticos [Küpfmüller, 1954].

Como Küpfmüller demuestra, al modelar secuencias de letras de textos en inglés y alemán usando un modelo de Markov de orden incremental, cadenas aleatorias pueden ser generadas de tal forma que su parecido con el lenguaje correspondiente es asombroso a pesar de su falta de significado en tales textos. Con esta idea básica, es solo un pequeño paso desde los grafemas en textos escritos a unidades fonéticas en el lenguaje hablado.

De forma similar a otras técnicas de identificación y reconocimiento, el modelado fonotáctico en LID involucra una fase de entrenamiento y una fase de prueba. Durante el entrenamiento, un conjunto de probabilidades es estimado por los datos de entrenamiento en cada lenguaje; en la fase de prueba, las probabilidades de observación de una muestra son calculadas dados todos los modelos, típicamente seguido por una regla de decisión de máxima verosimilitud. Los modelos probabilísticos fonotácticos pueden tener una variedad de estructuras. La más popular de estas es aquella de un modelo ergódico de orden de N-1, donde N = 1,2,…, también conocido como N-gramas [Hazen y Zue, 1997; Muthusamy et al., 1994; Zissman y Singer, 1994]. Técnicas alternativas para lograr un modelado más flexible de N-gramas incluyen árboles binarios de decisión [Navrátil, 2001] y varios esquemas de agrupamiento y suavizado [Kirchhoff et al., 2002b; Schukat-Talamazzini et al., 1995.]. A continuación se describe en la tabla 3.2 ejemplos de sistemas LID fonotácticos.

Tabla 3.2 - Ejemplos de sistemas LID fonotácticos y sus tasas de reconocimiento.

Sistema	Tarea	Duracion de la señal de prueba	Tasa de identificación	Referencia
Tokenizer multilingue de trigramas interpolados	OGI-11L	10s/45s	62.7%/77.5%	[Hazen y Zue, 1997]
Bigramas (1 tokenizer) dependientes del genero	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]
Modelado de flujos cruzados.	OGI-10L	Mezclados	65%	[Parandekar y Kirchoff, 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]

3.1.4 LID Prosódico.

Información prosódica-como el tono, entonación y prominencia- es codificada principalmente en dos componentes de la señal: la frecuencia fundamental (*F*0) y la amplitud. De este modo, propiedades de la *F*0 y contornos de amplitud pueden intuitivamente ser útiles en LID automático. En Eady [Eady, 1982], por ejemplo, dos lenguajes con diferentes propiedades prosódicas fueron estudiados: inglés, el cual pertenece a la categoría de los leguajes que marcan diferentes niveles de prominencia por medio de la *F*0 y amplitud, y el Chino, en el cual los patrones tonales son léxicamente distintivos. El autor comparó los contornos de tiempo de la frecuencia fundamental y la energía extraída de sentencias del mismo tipo, y encontró diferencias únicas específicas del lenguaje, tal como una mayor tasa de cambio en cada contorno y una fluctuación más alta entre silabas individuales para el chino.

Sin embargo, el uso de la prosodia para identificar lenguajes conlleva 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 especificas 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/función en discurso (énfasis contrastivo, dado contra nueva distinción)

En la tabla 3.3 se describen experimentaciones de algunos componentes prosódicos.

Tabla 3.3 - 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]

3.1.5 LID basado en LVCSR

El acercamiento LID más efectivo utiliza idealmente un conocimiento completo de la estructura léxica y gramatical de un lenguaje. El acercamiento automático más cercano a la situación ideal de un oyente humano familiarizado con un lenguaje dado es representado por sistemas LID usando grandes identificadores de habla de vocabulario continuo (LVCSR) para codificar una muestra entrante en cadenas de palabras con un subsecuente

análisis para patrones específicos LID. Las ventajas de este enfoque son obvias: debido a que restricciones de alto nivel pueden ser utilizadas, los resultados esperados son más precisos y con menos ruido que aquellos de un sistema fonotáctico o puramente acústico. Por otro lado, el requerimiento para que el LVCSR sea entrenado con datos propiamente transcritos en cada lenguaje, implica un considerable esfuerzo durante el entrenamiento- no sin mencionar que dichos recursos de entrenamiento podrían no estar disponibles para todos los lenguajes. Un sistema LID basado en LVCSR también requiere más recursos computacionales durante las pruebas, lo que lo hace una solución menos atractiva para aplicaciones en dispositivos de mano y portátiles.

De manera concreta, el principio básico de un sistema LID basado en LVCSR es un acercamiento de fuerza bruta; una muestra de prueba en un lenguaje desconocido es procesada usando todos los sistemas LVCSR disponibles en todos los lenguajes hipotéticos. Dado que se asume que cada componente produce un resultado normalizado expresando el grado al cual la hipótesis asemeja la señal, una decisión del resultado máximo se lleva a cabo sobre todas las salidas para seleccionar la hipótesis final de lenguaje. En la tabla 3.4 se puede observar algunos experimentos LVSCR con sus respectivas tasas de error.

Tabla 3.4 - Algunos componentes LID LVSCR y su tasa de error.

Sistema	Tarea	Duración de prueba	Tasa de identificación	Referencia
LVCSR	4L	<5s	84%	[Schultz et al., 1996]]
LVCSR	6L	10s	97%	[Hieronymus y Kadambe,1997]
LVCSR	2L (control Aéreo)	mezclado	98.4%	[Fernandez et al, 2004]

Se ha revisado como los enfoques principales pueden ser descritos en términos de las características de la señal que buscan explotar, así como en términos de sus técnicas de modelado. Sin embargo, los variados grados de dificultad en los experimentos LID, tal como la duración de la prueba, cantidad de datos de entrenamiento y anotación de datos, calidad del canal, hace difícil una comparación consistente de los algoritmos individuales. Basado en una comparación algo ruda, una categorización basada en desempeño pone a los

sistemas basados en LVCSR entre los métodos más poderosos, seguido del enfoque fono táctico, el acústico, y finalmente el prosódico.

A continuación en la tabla 3.5 se muestra una comparación de los enfoques principales para LID.

Tabla 3.5 - Comparación de los enfoques básicos LID desde una perspectiva de desarrollo de aplicación.

Enfoque Básico	Ventajas	Desventajas	Aplicación de ejemplo
Prosódico	Robusto en inconsistencia de canales	Mayormente usado para distinguir grupos de lenguajes	Preclasificador de lenguajes tonales contra lenguajes de estrés.
Acústico	Bajo costo de entrenamiento y pruebas (datos y computación)	Usable en combinación con otros componentes.	Identificación de lenguajes y acento en un sistema multienfoque.
Fono táctico	Buena tasa de desempeño/costo, no se necesita conocimiento lingüístico para entrenar	Útil para pruebas con duración >5 segundos	Sistemas LID con una población grande de lenguajes, incluyendo lenguajes raros sin datos de entrenamiento lingüísticamente etiquetados
LVCSR/reconocimiento de palabras clave	Alta precisión, pruebas cortas	Esfuerzo de entrenamiento significativo, entradas lingüísticas requeridas	Sistemas de dialogo multilingüe con componentes LVCSR, sistemas de minería de audio

3.2 Trabajos Relacionados con Wavelets

3.2.1 Reyes-Herrera

La línea de investigación de Reyes Herrera [Reyes-Herrera 2007] sigue la trazada por Cummins y Rouas, es decir, basados en modelado prosódico, la aportación de Reyes es que por primera vez en este tipo de estudios se introdujo la transformada Wavelet como forma de caracterizar la señal de habla, tiene como precedente la utilización en el reconocimiento del habla, en la detección de voz activa, reconocimiento del locutor, entre otras [Gupta y Gilbert 2001; Hioka y Hamada 2003].

La metodología usada por Reyes en este trabajo usa la transformada Wavelet Daubechies db2 con cuatro coeficientes y normalizados a [-1, 1]. El método se basa en la idea de que los coeficientes de mayor magnitud corresponden a una buena representación de la señal de voz, y los coeficientes con magnitudes pequeñas corresponden a una mala representación de la señal. Entonces el siguiente paso es hacer un truncado de los coeficientes para solamente obtener los coeficientes de mayor magnitud, de esta forma se truncan de acuerdo a su magnitud con un umbral del 1%, eliminando coeficientes que no representan bien la señal. Por ejemplo, para una muestra de 10 segundos donde obtuvo 131072 coeficientes, se hizo una reducción a 1312.

Finalmente, usando la base de datos OGI_TS, discriminando entre pares de idiomas con los mismos idiomas que Rouas usó en su experimentación, con excepción del Francés, compara sus resultados con los de él, por ejemplo, para una muestra de 50 segundos (Tabla 3.6) sus resultados muestran que el uso de la transformada wavelet es muy pertinente en esta clase de estudios, mejorando los resultados obtenidos por Rouas.

Tabla 3.6 - Resultados del clasificador usado por Reyes Herrera con muestras de 50 segundos

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Ingles	97	97	93	94	96	95	99	96
Alemán	-	93	94	93	98	98	94	91
Español	-	-	91	86	92	98	91	94
Mandarín	-	-	-	95	95	93	89	94
Vietnamita	-	-	-	-	93	96	95	95
Japonés	-	-	-	-	-	93	89	94
Coreano	-	-	-	-	-	-	95	91
Tamil	-	-	-	-	-	-		90

3.2.2 Vargas Martínez

[Vargas-Martínez 2008] sigue la misma línea de investigación que Reyes Herrera, el uso de la transformada Wavelet para caracterizar la señal de habla, el aporte de Vargas Martínez a la investigación de Reyes, es que él contempla cuestiones no vistas en el trabajo de Reyes, como lo es 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 necesita 50 segundos, mas el procesamiento). También se utilizó un modulo de detección de voz activa, para la eliminación de pausas largas. Reyes no usa medidas estadísticas sobre los coeficientes wavelet, como si lo hace Rouas sobre la F0.

De esta forma la metodología está diseñada para trabajar con muestras pequeñas de habla, además de eliminar pausas largas en las señales de habla, usa medidas estadísticas simples (media, desviación estándar, máximo, mínimo) sobre los coeficientes wavelet obtenidos a partir de la señal segmentada Se hace un truncado por fracción, para extraer los

coeficientes más representativos de la señal de habla, después se aplica la ganancia de información para posteriormente terminar en el modulo de identificación de lenguaje, donde se usa un clasificador Naive Bayes (Figura 3.1).

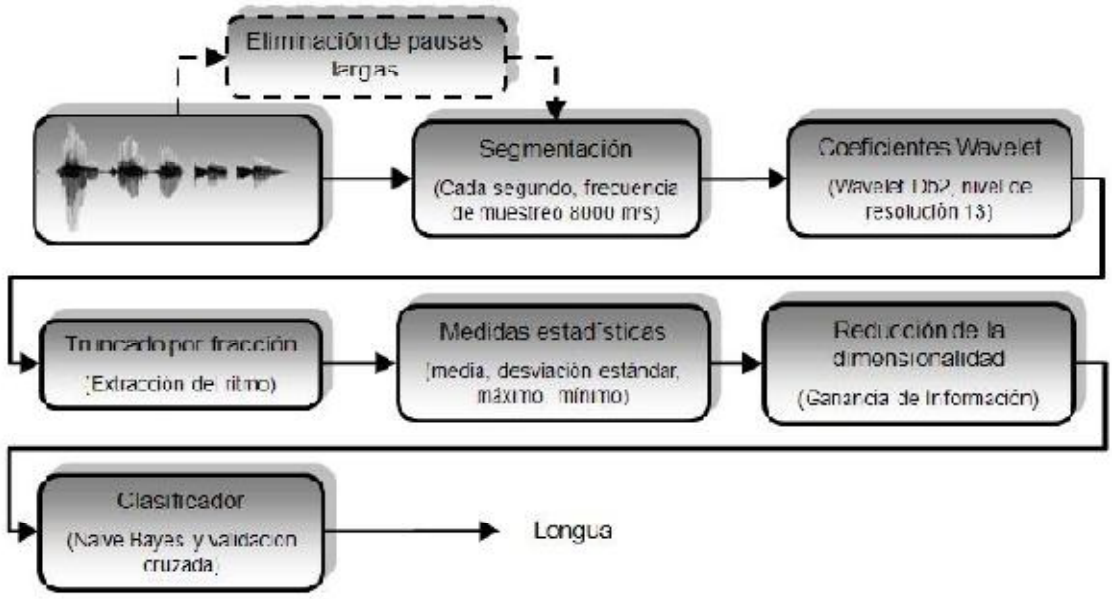


Figura 3.1 - Modelo del sistema de identificación de lenguas de Vargas Martínez.

Para las pruebas y experimentación se uso la base de datos telefónica OGI_TS, con habla espontánea, se tomaron 50 muestras por idioma, de una duración de 50 segundos de los siguientes 9 idiomas: Inglés, Alemán, Español, Mandarín, Vietnamita, Japonés, Coreano, Tamil y Farsi. Se realizaron pruebas con diferente duración en las muestras de habla: 30s, 10s, 5s y 4s. Dando como mejores resultados las de menor tiempo (Figura 39).

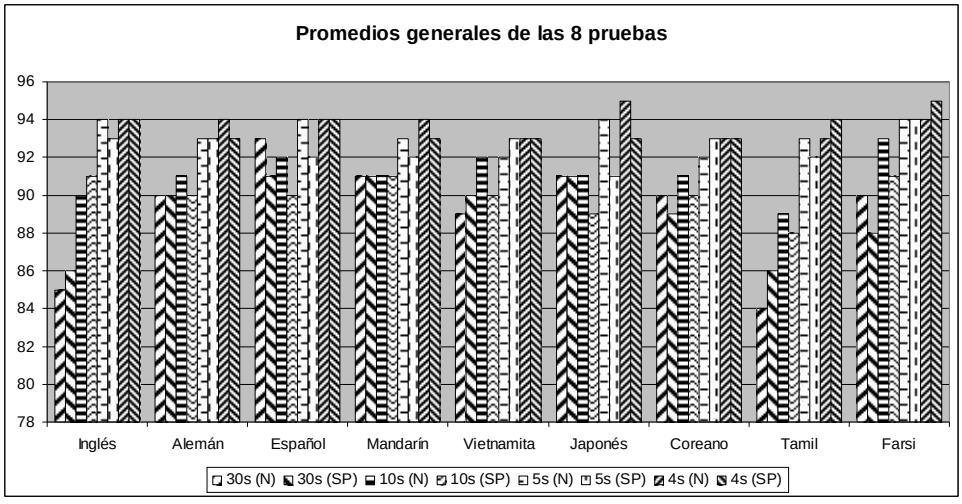


Figura 3.2 - Porcentajes de clasificación por idioma (Normal y sin pausas)

3.3 Análisis del estado del arte

Para resolver el problema de reconocimiento automático de idiomas, diferentes investigadores han empleado principalmente 4 acercamientos:

- Prosódico
- Acústico
- Fonotáctico
- LVCSR

Los investigadores que han usado la técnica de la transformada wavelet se han enfocado en el acercamiento puramente acústico, que es donde entra el presente trabajo, han usado biclasificadores para su experimentación y resultados, este trabajo propone una muticlasificación usando técnicas de torneo al usar un conjunto de biclasificadores para los torneos, un método que no ha sido explorada en el estado del arte.

Capítulo 4 Metodología

4.1 Biclasificadores

La metodología de solución está basada en la mostrada por [Vargas-Martínez 2008] quien sigue la misma línea de investigación que Reyes Herrera, el uso de la transformada Wavelet para caracterizar la señal de habla (Figura 4.1)

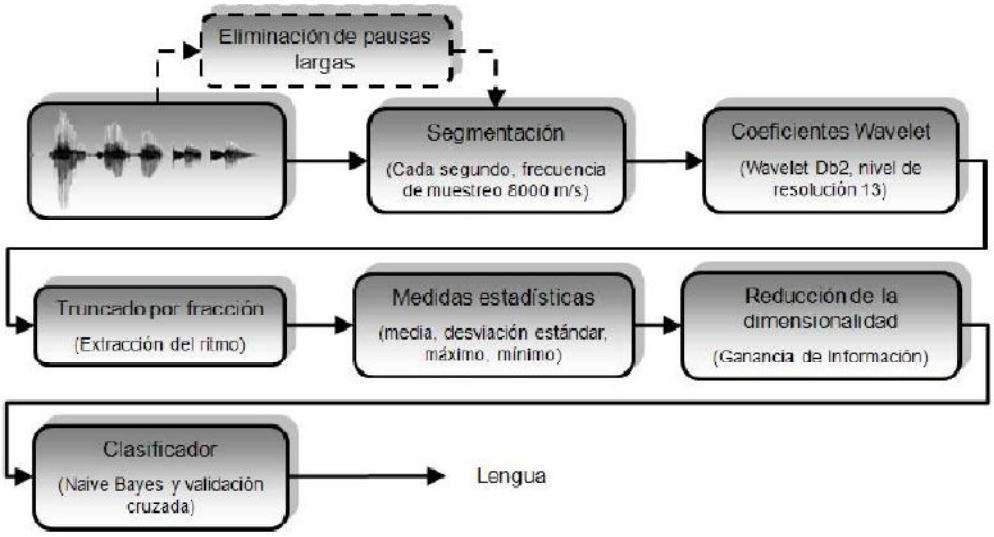


Figura 4.1 - Modelo del sistema de identificación de lenguas de Vargas Martínez.

4.1.1 Segmentación

El siguiente proceso se ha realizado con el software Praat, las imágenes siguientes han sido capturadas de dicho software.

En este paso se usa el software Praat, el cual es un software especializado en el análisis acústico.

En el proyecto se cuentan con la base de datos OGI_TS, que cuenta con archivos de 45 segundos, procedemos a realizar una segmentación del archivo, para obtener 45 archivos de un segundo, que si se mezclaran de forma continua, componen la muestra de sonido original. El primer paso para realizar esto es encontrar el archivo que queremos trabajar.

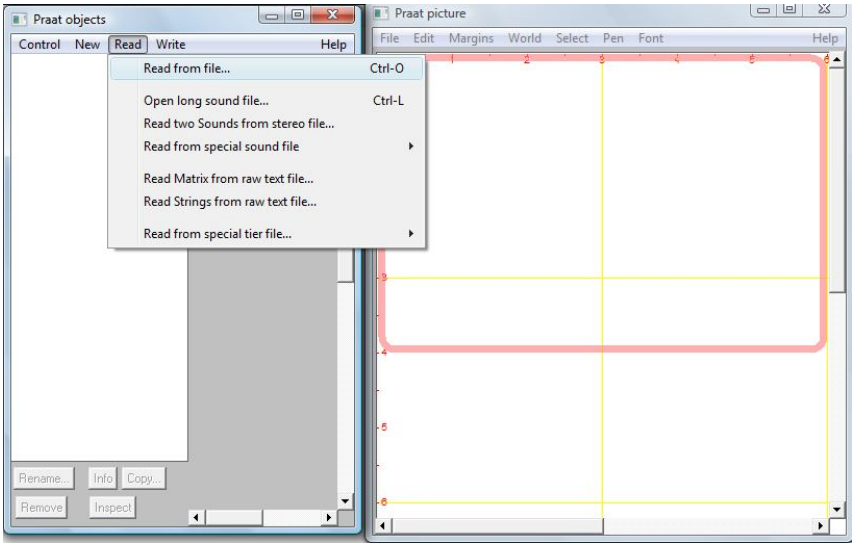


Figura 4.2 - Abriendo un archivo nuevo para procesar.

Una vez que se tiene al archivo abierto, este aparecerá en el explorador de objetos que utiliza Praat, para verificar que se haya agregado correctamente, y seleccionando el objeto procedemos al menú de edición del objeto (Fig. 4.3).

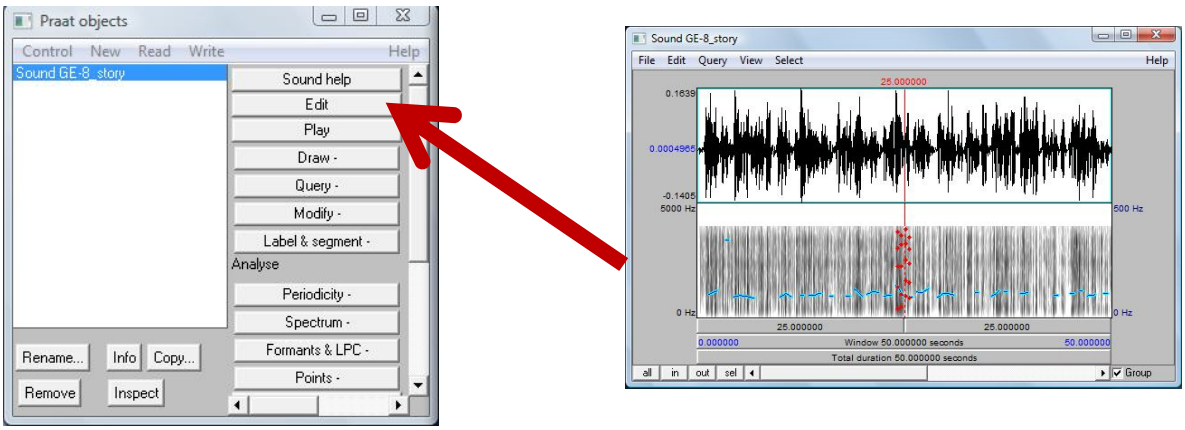


Figura 4.3 - Explorador de objetos de Praat y su ventana de edición.

Se procesa a elegir la segmentación, insertando los parámetros necesarios: el inicio y final del segmento deseado. En este caso como queremos dividir en segmentos de un segundo, por ejemplo, se elegiría empezar en el segundo 0 y terminar en el 1, posteriormente elegir 1-2, 2-3, 3-4… hasta terminar los 45 segmentos de un segundo. Estos parámetros se insertan en la ventana de selección, mostrada en la Fig.4.4

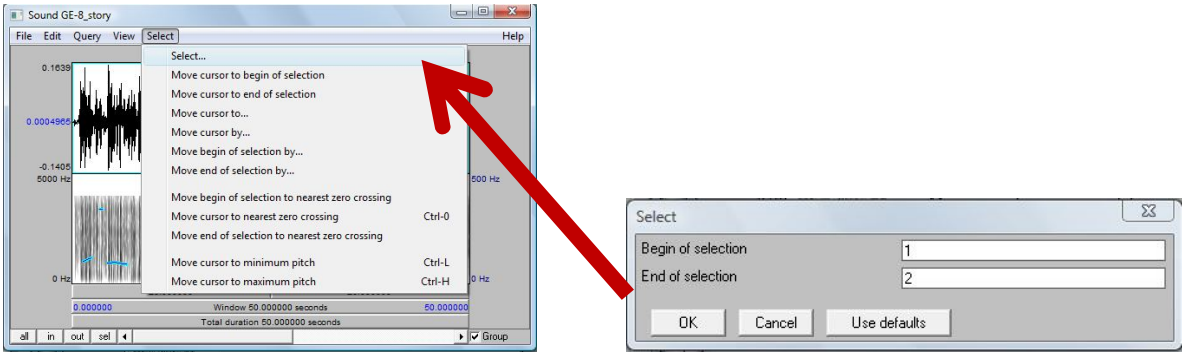


Figura 4.4 - Ventana de selección de segmentos.

Para verificar que el proceso se está realizando correctamente, en la ventana de edición, se marca con rojo el segmento elegido como se ve en la Fig.4.5 , cada vez que un segmento es elegido, se tiene que guardar dicho segmento, usando la instrucción que se muestra en la Fig.4.6

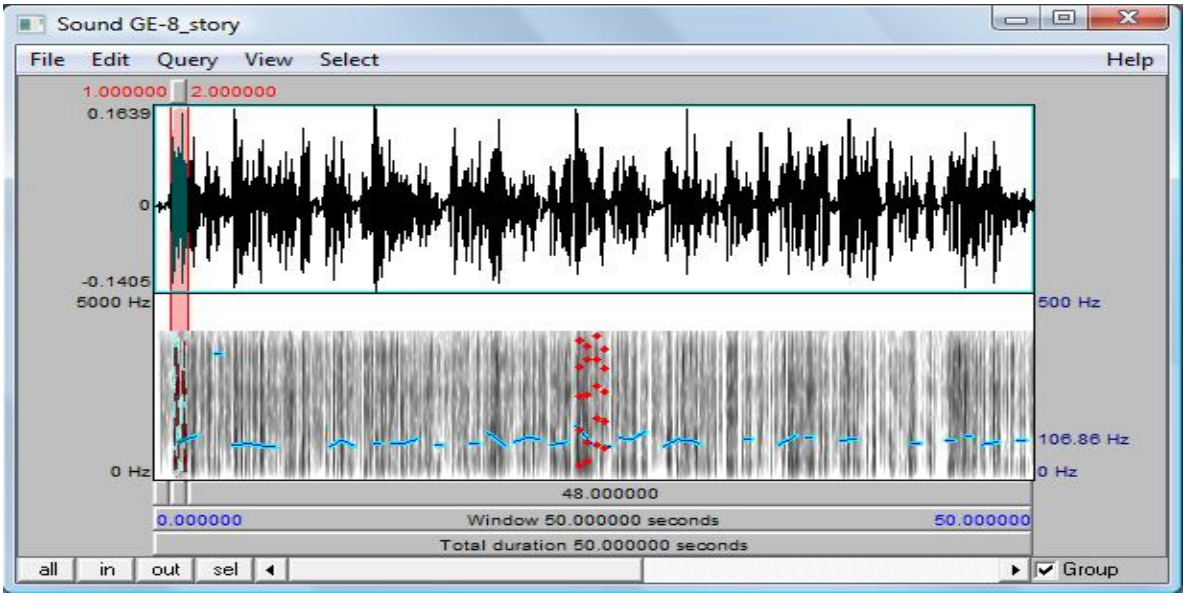


Figura 4.5 - Un segmento elegido marcado con rojo.

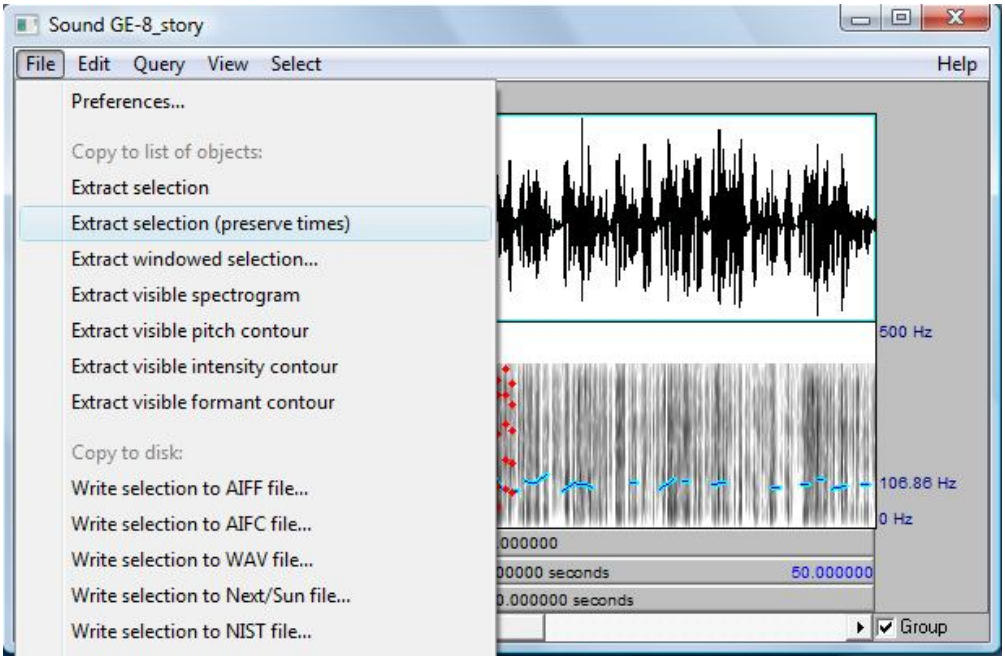


Figura 4.6 - Instrucción Extract selection (preserve times) para guardar el segmento nuevo.

4.1.2 Coeficientes Wavelet

Para cada segmento de un segundo, se aplica la transformada Wavelet Db2, este proceso se sigue realizando con Praat, los resultados obtenidos se verificaron con Matlab, y se concluyó que eran equivalentes. Para realizar este proceso en Praat, se utilizan los archivos de 1 segundo del paso anterior, que aparecen en el explorador de objetos después de que se usa la instrucción Extract selection (preserve times), se pasa a seleccionar el nuevo objeto y se utiliza el menú spectrum, donde se puede selecciona la opción wavelet, para posteriormente elegir el numero de coeficientes que se quieren (Fig. 4.7).

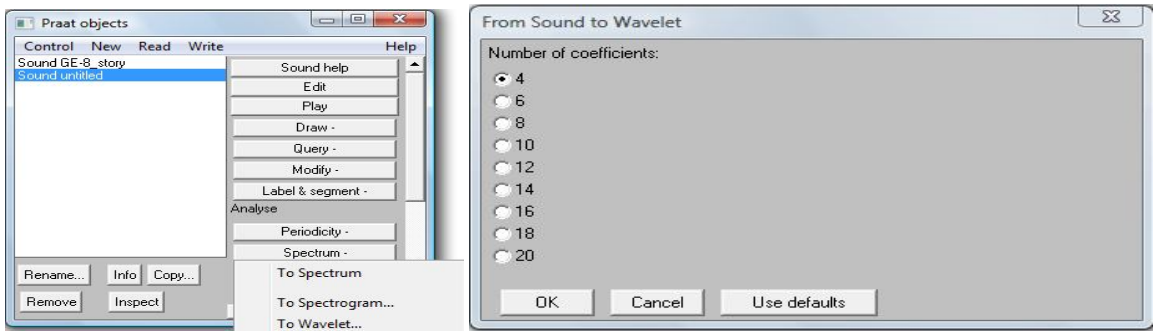


Figura 4.7 - Menú para seleccionar los coeficientes wavelet.

4.1. 3 Truncado por fracción

Al tener un nuevo objeto en el explorador, ahora de tipo wavelet, se puede observar que las opciones del menú cambian, y claramente vemos una nueva opción llamada truncate by fraction, como se muestra en la Fig. 4.8, en esta nueva opción se elige el porcentaje por el que se quiere fraccionar, eso se realiza debido a que se ha demostrado que los coeficientes más significativos, donde está la información más esencial, son los primeros, es por eso que se hace el truncado al 1%.

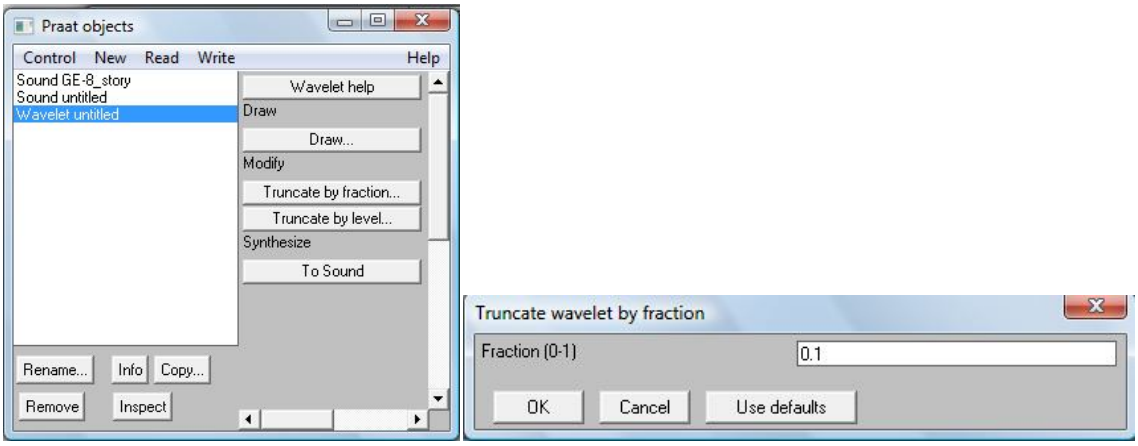


Figura 4.8 - Truncado por fracción.

Por último, después de todo el proceso que comenzó con la segmentación, falta guardar todo lo realizado, esto se realiza con la instrucción Write to text file… como se realiza en la Fig. 4.9

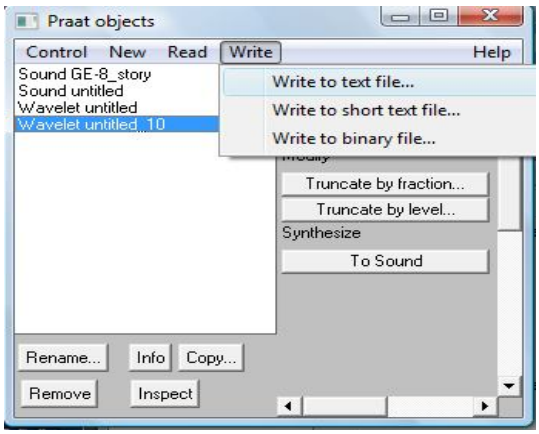


Figura 4.9 - Guardar todos los cambios realizados hasta el momento

4.1.4 Medidas estadísticas

Para realizar las medidas estadísticas, se utiliza el software Matlab, con este software a partir de un código, se transforman los archivos .wavelet del paso anterior, en nuevos archivos que contienen información estadística (media, desviación estándar, máximo y mínimo), en esta parte del proceso se harán diferentes experimentaciones con otras medidas estadísticas diferentes, se harán pruebas con la curtosis así como el skewness.

4.1.5 Archivos weka

Se crean los archivos .arff que maneja Weka, con el software Matlab, se utilizan los archivos de información estadística que se obtuvieron en el paso anterior, para darle un formato que pueda reconocer Weka, se hace mediante un código programado en Matlab, donde se escriben los encabezados y formato necesario a los archivos estadísticos para obtener el formato de Weka, y se guardan en su formato respectivo .arff.

4.1.6 Clasificación

Se realiza la clasificación usando el software Weka, este se utiliza usando la línea de comandos de Weka a través de una interfaz programada en Java, específicamente se usa el entorno de programación NetBeans, debido a que Weka utiliza mucha memoria del sistema debido a que los archivos son muy grandes, y otros entornos de programación Java diferentes, han causado conflictos. Se utiliza el clasificador NaiveBayes.

4.2 Clasificación por torneos.

Con combinaciones de clasificadores binarios, los métodos de clasificación por torneos se enfocan en hacer una clasificación mediante un proceso de decisión parecido a un torneo, donde las diferentes clases compiten contra cada una para producir la clase ganadora.

Las clasificaciones por torneos más populares, son las modeladas en base a reglas de deportes y juegos, he de ahí de donde toman su nombre y sus reglas, torneo de eliminación y torneo round robin.

El método de eliminación por torneo basa sus reglas en las reglas de competencia del campeonato de Wimbledon. En esta competencia, cada jugador debe competir con los otros jugadores que se determinaron en un sorteo antes de que el evento comience. Cuando termina un partido, el ganador pasa a la siguiente ronda y el perdedor es eliminado. Se genera un sorteo antes de la clasificación para generar las reglas de competencia. La clase ganadora en el último round es la clase óptima.

4.2.1 Método por eliminación

Se crea un vector con todos los idiomas a clasificar.

A	E	F	I	J	K	M	T	V

Inicializar el torneo de forma aleatoria, al generar números aleatorio entre 0 y el último índice de la matriz, para ir introduciendo a un vector nuevo los idiomas de forma aleatoria. Se hace hasta que se haya visitado cada elemento del vector, si se llega a repetir el número aleatorio, se repite hasta encontrar un número no repetido.

Ejemplo:

Vector inicial

A	E	F	I	J	K	M	T	V
0	1	2	3	4	5	6	7	8

Aleatorio: 1

Vector inicial:

A	X	F	I	J	K	M	T	V
0	1	2	3	4	5	6	7	8

Vector torneo:

E								
0	1	2	3	4	5	6	7	8

Aleatorio: 8

Vector inicial:

A	X	F	I	J	K	M	T	X
0	1	2	3	4	5	6	7	8

Vector torneo:

E	V							
0	1	2	3	4	5	6	7	8

Aleatorio: 5

Vector inicial:

A	X	F	I	J	X	M	T	X
0	1	2	3	4	5	6	7	8

Vector torneo:

E	V	K						
0	1	2	3	4	5	6	7	8

Así sucesivamente se repite el proceso, hasta tener en nuestro Vector torneo todos los idiomas. A continuación se dará un vector torneo con el que se ejemplificará el proceso.

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

Al tener listo el sorteo, se puede empezar a realizar el torneo de eliminación, como en este tipo de torneos se necesita un número par, antes de empezar el torneo, se hace el "partido" con los dos primeros elementos del vector y el ganador se agrega al final, de, esta forma tenemos un número para empezar el torneo. En nuestro vector torneo de ejemplo, se realiza el biclasificador E-V y el ganador se agregará al final.

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

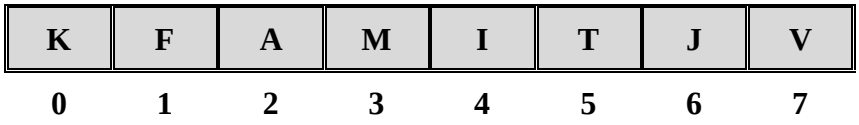
Vector fase siguiente:

K	F	A	M	I	T	J	V
0	1	2	3	4	5	6	7

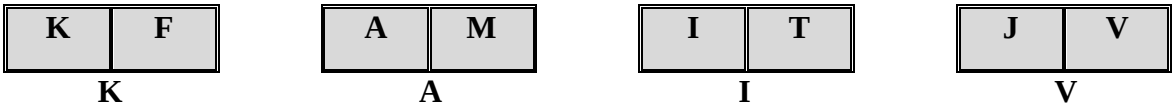
Ya se puede empezar un torneo de eliminación, agarrando cada dos idiomas consecutivos del vector para hacer el biclasificador, siempre se usaran dos vectores, el vector torneo y el vector siguiente que almacena la siguiente fase, como un torneo de eliminación, serian cuartos de final, semifinal y final. El idioma ganador será al grupo al que pertenece la instancia a clasificar.

Ejemplo del proceso:

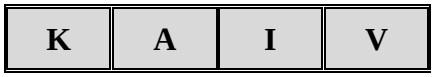
Vector torneo:



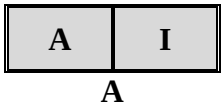
Resulta en la siguiente competencia;



Vector siguiente:



Vector siguiente:



El ganador del último encuentro se considera la clase perteneciente de la instancia de entrada a clasificar.

De manera resumida, se puede ver el proceso de eliminación en esquema mostrado en la figura 4.10.

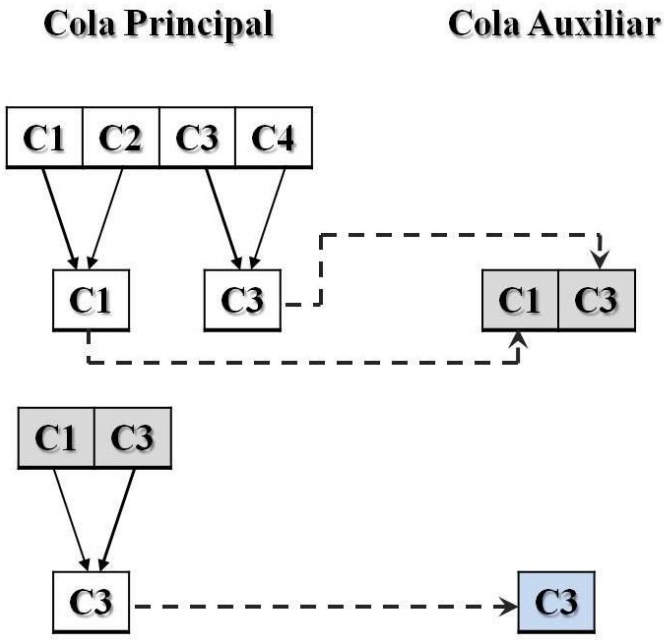


Figura 4.10 - Ejemplo de método de eliminación.

4.2.2 Torneo Round Robin

Este método consiste en tener un grupo, y que se hagan todas las combinaciones posibles de biclasificadores, y se les dé un puntaje dependiendo de la clase ganadora, entonces al final, la clase que haya obtenido más puntos será el idioma al que pertenece la instancia.

4.2.2.1 Variante con dos grupos

En este caso se opta por hacer dos grupos, un grupo que tenga 5 idiomas y otro grupo que tenga 4 idiomas, el proceso de selección de dichos grupos es de forma aleatoria.

Se crea un vector con todos los idiomas a clasificar.

A	E	F	I	J	K	M	T	V

Esta parte del proceso es similar al método anterior, hasta el punto en que se tiene un vector aleatorizado.

El siguiente será un vector aleatorizado con el que se ejemplificará:

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

Cuando se tiene el vector torneo aleatorizado, se pasa a formar los grupos a participar, dividiendo dicho vector en 2 partes, uno con los primeros 5 elementos y otro con los restantes.

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

Grupo 1:

E	V	K	F	A
0	1	2	3	4

Grupo 2:

M	I	T	J
0	1	2	3

Ya que se tienen los dos grupos, se procede a realizar un proceso de puntaje, al hacer todas las combinaciones posibles en cada grupo, de tal forma que si gana una clase se le un puntaje a favor de 3 puntos, de lo contrario no obtiene puntos. Teniendo nuestros 2 vectores de grupo como ejemplo se tiene:

Grupo 1	E	V	K	F	A	Puntaje
E	-	3	3	0	3	9
V	3	-	3	0	0	6
K	3	3	-	0	0	6
F	0	0	0	-	3	3
A	3	0	0	3	-	6

Grupo 2	M	I	T	J	Puntaje
M	-	3	0	3	6
I	3	-	3	3	9
T	0	3	-	0	3
J	3	3	0	-	6

Al tener los mejores puntajes, se juntan los dos idiomas para ejecutar un último biclasificador que se forma por dichos idiomas, donde el ganador será la clase asignada a la

instancia de entrada, en este caso el idioma ganador del grupo 1 es Español y del grupo 2 es Inglés, así que se ejecutaría el biclasificador E-I.

Ganador Grupo 1	Ganador Grupo 2

Ganador General

4.2.2.3 Variante con tres grupos

Esta variante usa tres grupos, donde cada uno está compuesto por tres idiomas diferentes, el proceso para conformar dichos grupos se hace de forma aleatoria.

Se crea un vector con todos los idiomas a clasificar.

A	E	F	I	J	K	M	T	V

Esta parte del proceso es similar al método anterior, hasta el punto en que se tiene un vector aleatorizado.

El siguiente será un vector aleatorizado con el que se ejemplificará:

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

Cuando se tiene el vector torneo aleatorizado, se pasa a formar los grupos a participar, dividiendo dicho vector en 3 partes, los primeros 3 un grupo, los siguientes 3 el segundo grupo y los últimos 3 el tercer grupo.

Vector torneo:

E	V	K	F	A	M	I	T	J
0	1	2	3	4	5	6	7	8

Grupo 1:

E	V	K
0	1	2

Grupo 2:

F	A	M
0	1	2

Grupo 3:

I	T	J
0	1	2

Ya que se tienen los tres grupos, se procede a realizar un proceso de puntaje, al hacer todas las combinaciones posibles en cada grupo, de tal forma que si gana una clase se le un puntaje a favor de 3 puntos, de lo contrario no obtiene puntos. Teniendo nuestros 3 vectores de grupo como ejemplo se tiene:

Grupo 1	E	V	K	Puntaje
E	-	3	3	6
V	3	-	0	3
K	3	0	-	3

Grupo 2	F	A	M	Puntaje
F	-	3	0	3
A	3	-	3	6
M	0	3	-	3

Grupo 3	I	T	J	Puntaje
I	-	0	3	3
T	0	-	3	3
J	3	3	-	6

Al tener los mejores puntajes, se crea un nuevo grupo con dichos puntajes, para repetir el proceso, donde el ganador de este grupo será la clase asignada a la instancia de entrada, en este caso los idiomas ganadores fueron español, alemán y japonés, así que se crea un grupo nuevo.

Grupo Final	E	A	J	Puntaje
E	-	3	3	6
A	3	-	0	3
J	0	3	-	3

El ganador final, será la clase con más puntos.

Capítulo 5 Experimentación y Resultados

5.1 Pruebas

A lo largo de este periodo se realizaron una extensa serie de pruebas, que en los siguientes apartados se explicarán más detalladamente. Se realizaron pruebas con múltiples clasificadores para justificar el porqué del uso de Naive Bayes, que fue el clasificador que mejores resultados obtuvo.

Las pruebas se hicieron en un equipo con las siguientes características:

- Sistema Operativo: Windows 7 - 64 bits
- Memoria Ram: 4 Gigas
- Procesador: Core 2 Duo 7300

5.1.1 Justificación del uso de Naive Bayes

Se hicieron pruebas con diferentes clasificadores, se muestran los promedios de la clasificación de cada idioma con el respectivo clasificador. En la tabla 5.1 se muestra que los resultados del clasificador NaiveBayes están por encima de los otros clasficiadores de la prueba.

Tabla 5.1 - Prueba de 30 segundos

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Ibk	64	71	74	70	74	71	74	68	73
J48	67	72	71	70	72	71	74	66	72
Lmt	77	81	80	80	79	80	79	77	78
LWL	52	53	51	54	54	54	53	56	55
NaiveBayes	85	90	93	91	89	91	90	84	90

Tabla 5.2 - Prueba de 10 segundos

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Ibk	72	73	78	74	72	73	76	75	75
J48	72	74	72	73	71	74	70	70	72
Lmt	77	80	79	79	80	82	77	80	78
LWL	55	56	54	55	53	56	53	55	52
NaiveBayes	90	91	92	91	92	91	91	90	93

Tabla 5.3 - Prueba 5 Segundos

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Ibk	73	77	77	77	78	78	79	77	80
J48	74	75	73	73	76	77	72	75	73
Lmt	84	82	81	83	82	81	79	82	82
LWL	60	59	55	58	55	59	57	57	58
NaiveBayes	94	93	94	93	92	94	92	93	95

Tabla 5.4 - Prueba 4 Segundos

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Ibk	78	78	78	78	75	82	76	76	79
J48	74	73	76	77	75	76	73	74	76
Lmt	81	81	81	81	81	81	81	81	81
LWL	60	59	57	57	55	55	59	58	57
NaiveBayes	94	94	93	94	93	95	93	93	94

5.1.2 Uso de nuevas medidas estadísticas

Uno de los objetivos que se habían planteado, era el incorporar nuevas medidas estadísticas con el fin de mejorar los resultados, se incorporaron dos nuevas medidas estadísticas, la curtosis y la skewness, se lograron obtener resultados mejores a los que se tenían, se hicieron pruebas incluyendo solo la curtosis, incluyendo también solo la skewness y cuando se incluyen las dos.

5.1.2.1 Pruebas de 30 segundos

Tabla 5.5 - 30 Segundos con medidas estadísticas anteriores.

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	84	88	83	80	87	83	81	90
Alemán		94	93	90	93	93	84	88
Español			95	94	96	96	92	89
Mandarín				99	93	92	84	92
Vietnamita					92	92	76	89
Japonés						90	85	94
Coreano							86	91
Tamil								83

Tabla5.6 - 30 SegundoS agregando la medida estadística Skewness

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	87	93	86	84	90	87	84	92
Alemán		97	95	95	94	93	89	97
Español			95	96	98	98	96	94
Mandarín				99	97	92	87	95
Vietnamita					95	96	78	97
Japonés						93	90	96
Coreano							94	93
Tamil								89

Tabla 5.7 - 30 Segundos agregando la medida estadística Curtosis.

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	84	88	85	81	87	83	83	91
Alemán		94	95	93	97	92	84	93
Español			97	94	97	98	93	90
Mandarín				99	95	90	84	95
Vietnamita					92	93	76	93
Japonés						95	86	94
Coreano							92	92
Tamil								85

Tabla 5.8 - 30 Segundos agregando Curtosis y Skewness.

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	87	93	88	85	90	87	87	94
Alemán		97	94	95	96	92	89	97
Español			97	96	98	99	96	96
Mandarín				99	98	94	87	96
Vietnamita					94	97	79	98
Japonés						95	89	96
Coreano							95	92
Tamil								89

Tabla 5.9 - Promedios de las pruebas de 30segundos

Promedios									
	Inglés	**Alemán**	**Español**	**Mandarín**	**Vietnamita**	**Japonés**	**Coreano**	**Tamil**	**Farsi**
30s N	85	90	93	91	89	91	90	84	90
30s K	85	92	94	93	90	93	92	85	92
30s S	88	93	96	93	93	94	93	88	94
30s S-K	89	93	97	94	93	95	94	89	95

5.1.2.2 Pruebas de 5 segundos

Tabla 5.10 - 5 Segundos con medidas estadísticas anteriores

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	94	96	90	96	95	88	95	100
Alemán		94	93	92	93	95	94	92
Español			93	92	93	95	95	93
Mandarín				89	96	91	93	95
Vietnamita					91	92	91	95
Japonés						95	92	95
Coreano							87	95
Tamil								93

Tabla 5.11 - 5 Segundos agregando la medida estadística Skewness.

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

Tabla 5.12 - 5 Segundos agregando la medida estadística Curtosis.

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	96	97	91	97	97	90	97	100
Alemán		95	93	94	93	95	94	92
Español			93	93	93	95	95	93
Mandarín				89	96	92	94	94
Vietnamita					92	92	92	95
Japonés						93	91	95
Coreano							90	95
Tamil								95

Tabla 5.13 - 5 segundos agregando Curtosis y Skewness.

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

Tabla 5.14 - Promedios de las pruebas de 5 segundos

Promedios									
	Inglés	**Alemán**	**Español**	**Mandarín**	**Vietnamita**	**Japonés**	**Coreano**	**Tamil**	**Farsi**
5s N	94	93	94	93	92	94	92	93	95
5s K	96	94	94	93	93	94	93	94	95
5s S	97	97	96	95	96	96	95	97	96
5s S-K	97	96	96	96	96	96	95	97	96

5.1.3 Uso de otros wavelet.

Otro objetivo que se tenía fue el de usar otra transformada wavelet para ver el comportamiento de los resultados de las pruebas, se siguió usando la familia Dbn de wavelets, donde se experimentó con Db4 y Db12. Los resultados obtenidos por estos nuevos wavelets no resultan en una mejora significativa a los resultados ya presentados.

Tabla 5.15 - Promedios de la prueba de 10 segundos para diferentes wavelet

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Db2	95	96	96	95	97	95	95	94	96
Db4	93	95	95	94	95	93	96	91	96
Db6	95	95	94	93	96	94	94	92	95
Db10	94	94	95	94	94	95	93	92	96

5.2 Multiclasificación

Teniendo ya establecida la metodología para la multiclasficación, se realizaron diversas pruebas con las variantes de las técnicas propuestas, cabe señalar que aunque los resultados son muy prometedores, aun hay que trabajar en el método, ya que hay 2 idiomas que salen relativamente bajos, y el tiempo que tarda para la clasificación es alto. A

continuación se muestran los resultados de los experimentos, donde ya se ocupan las nuevas medidas estadísticas al observar que mejoran los resultados con los experimentos anteriores.

Tabla 5.16 - Promedios de las pruebas de multiclasificación

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Eliminación	82	100	96	100	98	100	98	74	96
Round Robin (2 Grupos)	80	100	94	100	98	100	98	72	98
Round Robin (3 Grupos)	80	100	94	100	98	100	98	74	96

Se encontraron dos investigadores que realizan multiclasificación, pero los resultados no son del todo comparables, debido a que la caracterización de la señal es diferente, los métodos empleados son diferentes e incluso tienen un acercamiento distinto, ya que el usado en este trabajo es puramente acústico, sin embargo a continuación se mencionaran:

Diamantino Caseiro propone una metodología únicamente fonotáctica basada en multiclasificación, utilizando la base de datos telefónica de lenguajes europeos *Speech-Dat-M*. Los resultados son presentados en la tabla 5.17.

Tabla 5.17 – Matriz de confusión del sistema de Caseiro.

	Inglés	Español	Alemán	Portugués	Italiano	Frances
Inglés	**81.40%**	0.40%	11.10%	1.70%	2.70%	2.80%
Español	1.90%	**70.60%**	2.30%	3.70%	15.90%	5.60%
Alemán	8.60%	1.50%	**82.40%**	1.20%	2.00%	4.20%
Portugués	2.50%	3.10%	1.90%	**87.80%**	1.70%	3.10%
Italiano	4.90%	14.40%	4.10%	1.10%	**70.00%**	5.60%
Francés	2.10%	2.90%	4.30%	1.20%	4.00%	**85.5%**

Torres-Carasquillo, en su trabajo, utiliza la base de datos OGI_TS, donde propone un modelo que no requiere reglas léxicas ni gramaticales, en sus resultados menciona que este modelo cuenta con una tasa de error de 29% en muestras de 10 segundos y 24% en muestras de 45 segundos. [Torres-Carrasquillo et al-2002].

5.3 Comparación de resultados (biclasificadores)

Se presentan los resultados obtenidos, resumidos en una tabla de promedios, donde se comparan con los resultados obtenidos por Reyes Herrera y Vargas Martínez. Las muestras tomadas son de 5 segundos.

Tabla 5.188 - Tabla comparativa de resultados de los biclasificadores.

	Inglés	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Reyes	90	84	86	87	85	86	85	85	86
Vargas	94	93	94	93	92	94	92	93	94
Propuesto	97	96	96	96	96	96	95	97	96

Como se observa en la tabla 5.18, los resultados obtenidos están por encima de los resultados del estado del arte.

Capitulo 6 Conclusiones y trabajos futuros

En el presente trabajo se realizaron un conjunto de experimentos, donde las conclusiones a estos se estructuran de la siguiente manera:

- Eleccion del clasificador
- Uso de otrass medidas estadísticas
- Número de atributos.
- Usó de otros wavelets de la familia Dbn
- Metodologia de multiclasificacion.

Se hizó un conjunto de experimentaciones con muestras de diferente tamaño (3, 4, 5, 10 y 30 segundos) para distintos clasificadores, donde Naive Bayes muestra un porcentaje de clasificación correcta por encima del 90%, mientras que los otros tienen una tasa de clasificación mucho mas baja. Es por esto que se opta por usar este clasificador. (Ver sección 5.1.1)

El agregar las nuevas medidas estadísticas (curtosis, skewness), incrementa el desempeño obtenido en los biclasificadores de Vargas, incrementando de 93% a 96% el porcentaje de clasificación correcta, incluso el solo añadir una de las dos medidas estadísticas nuevas, ya sea curtosis o skewness, mejora el promedio de clasificación, individualmente, el skewness es el que tiene mayor impacto positivo en los resultados. (Ver sección 5.1.2 y 5.2)

Al tener las nuevas medidas estadísticas añadidas, se tiene un número mayor de atributos en los biclasificadores, y aún asi tienen un mejor desempeño, mostrando que no precisamente al tener menos atributos tiene que haber una mejor clasificación.

Usando la familia de wavelets Dbn, con n>2, no mejora los resultados obtenidos en Vargas, como se puede observar en los resultados, esto indica que para este tipo de problema la familia Dbn tiene un comportamiento similar. (Ver sección 5.1.3)

Los resultados obtenidos en la multiclasificación usando técnicas de torneo son muy aceptables, a excepción de la clasificación del idioma Tamil, lo que indica que es una lengua complicada de clasificar, el tiempo de ejecución de estas técnicas es muy elevado, pero los resultados obtenidos son alentadores, clasificando las instancias de algunos idiomas con un 100% de clasificación correcta, se concluye que es una buena técnica de clasificación, que debe de seguir mejorando. (Ver sección 5.2)

6.1 Aportaciones

Este proyecto tiene las siguientes aportaciones:

Aportó una experimentación en la metodología agregando nuevas medidas estadísticas para la caracterización de la entrada de audio, así como una experimentación con diferentes wavelet de la familia Dbn.

Otra aportación es el diseño de una metodología de multiclasificación basada en los biclasificadores, que por lo resultados se indica viable seguir explorándolo, además de que no es un método que haya sido explorado por muchos investigadores.

6.2 Trabajos Futuros

Un punto a considerar es aumentar el número de idiomas a clasificar en los archivos weka, en las pruebas se oberva que existe una buena clasificación a pares, podría aumentar el número de idiomas a tres en los archivos .arff y observar el comportamiento de los resultados.

Se pueden hacer cambios en elegir otro tipo de wavelet para la metodología de creación de los biclasificadores, se ha experimentado con la familia de wavelets Dbn, se puede buscar otra familia wavelet diferente para experimentar con ella.

Otro punto importante es seguir trabajando en la multiclasificación, se observan resultados muy aceptables, pero el tiempo de ejecución de esta técnica es muy alto, debido a todas las combinaciones que se tienen que realizar con cada entrada a clasificar, se debe de trabajar en reducir el tiempo de ejecución, al utilizar diferentes métodos de selección de atributos. También se debe considerar implementar nuevos métodos de torneo para la multiclasificación.

La meta a futuro de este proyecto es poder usarse en las lenguas indígenas del país, por lo que se debe trabajar en una métodologia para construir una base de datos con las diferentes lenguas existentes, estandarizar las muestras basándose en el estándar mostrado en la OGI_TS.

Investigar otros métodos de minería de datos que se puedan aplicar a los coeficientes wavelet con el fin de reducir el número de atributos necesitados para representar el ritmo.

Referencias

[Berkling 1996] Berkling Kay Margarethe, Automatic Languaje Identification with Sequences of Language-Independent Phoneme Clusters, Tesis Doctoral, Oregon Graduate Institute of Science and Technology, 1996.

[Burrus et al. 1997] Burrus C. Sidney, Gopinath Ramesh A., Guo Haitao, Introduction to Wavelets and Wavelet Transform, Prentice Hall, 1997.

[Campbell et al. 2006] Campbell William, Gleason Terry, Navrátil Jiri, Reynolds Douglas, Shen Wade, Singer Elliot, and Torres-Carrasquillo Pedro, "Advanced Language Recognition using Cepstra and Phonotactics: MITLL System Performance on the NIST 2005 Language Recognition Evaluation", In IEEE Odyssey 2006: The Speaker and Language Recognition Workshop, (San Juan, Puerto Rico), June 2006.

[Caseiro y Trancoso 1998] Caseiro D., Trancoso I., "Language Identification Using Minimum Linguistic Information", 10th Portuguese on Pattern Recognition (RECPAD'98), Lisbon Portugal, 1998.

[Cimarusti e Ives 1982] Cimarusti D. and Ives R. B., "Development of An Automatic Identication System of Spoken Languages": Phase 1, In proceedings 1982 IEEE International Conference on Acoustics, Speech, and Signal Processing, Paris, France, May 1982.

[Cumins et al. 1999] Cummins, Fred, Felix Gers , Jürgen Schmidhuber, "Language Identification from Prosody Without Explicit Features", EUROSPEECH-99, pp. 371-374, 1999.

[Foil 1986] Foil J. T., "Language Identification Using Noisy Speech", In Proceedings 1986 IEEE International Conference on Acoustics, Speech, and Signal Processing, Tokyo, Japan, 1986.

[Goodman et al. 1989] Goodman F.J., Martin A.F., and Wohlford R.E., "Improved Automatic Language Identification in Noisy Speech". In Proceedings 1989 IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow, Scotland, May 1989.

[Gupta y Gilbert 2001] Gupta M., Gilbert A., "Robust Speech Recognition using Wavelet Coefficient Features", ASRU '01, IEEE, pp. 445- 448, 2001.

[Hioka y Hamada 2003] Hioka Yusuke, Hamada Nazomu, "Voice Activity Detection with Array Signal Processing in the Wavelet Domain", IEICE TRANSACTIONS on Fundamentals of Electronics, Vol E86-A, No 11, 2003.

[House y Neuberg 1977] House A. S. and Neuberg E. P., "Toward Automatic Identication of the Language of An Utterance, I. Preliminary methodological considerations. Journal of the Acoustical Society of America, 62(3):708-713, 1977.

[Jaffard et al.] Jaffard Stéphane, Meyer Ives, Ryan D. Robert, Wavelets Tools for Science and Technology, SIAM, 1962.

[Jansen-2005] Marten Jansen, Patrick Oonincx(2005).Second Generation Wavelets and Applications, Springer.

[Lee 2000] Edward A. Lee, Pravin Varaiya, Structure and interpretation of signals and systems, 2000. Addison Wesley.

[Li-2002] A Survey on Wavelet Applications in Data Mining ,2002, Tao Li, Qi Li, Shenghou Zhu, Mitsunomi, Ogihara, ACM SIGKDD Explorations. Volume 4, Issue 2-page 49

[Li y Edwards] Li K.P. and Edwards T. J., "Statistical models for automatic language identication", In Proceedings 1980 IEEE International Conference on Acoustics, Speech, and Signal Processing, Denver, CO, April 1980.

[**Misiti et al. 2010**] Wavelet Toolbox™ 4 User's Guide

[Michel-2004] Misiti Michel, Misiti Yves, Oppenheim Georges, Poggi Jean-Michel, Matlab7-Wavelet Toolbox (2004). Wavelets: A new tool for signals analysis.

[Mix and Olejniczak,2003] D. F. Mix and K. J. Olejniczak, *Elements of Wavelet for Engineers and Scientists*. Hoboken NJ, USA, Wiley-Interscience, 2003.

[Muthusamy 1992] Muthusamy Yeshwant K., "A Review of Research in Automatic Language Identification", Technical Report No. CS/E 92-009, Center for Spoken Language Understanding, Oregon Graduate Institute, 1992.

[Muthusamy et al. 1992] Muthusamy Yeshwant, Berkling Kay, Arai Takayuki, Cole Ronald, Barnard Etienne, "A Comparison of Approaches to Automatic Language Identification Using Telephone Speech", In EUROSPEECH'93, 1307-1310, 1993.

[Navrátil y Zühlke 1998] Navrátil, J. Zühlke W., "An efficient phonotactic-acoustic system for language identification", Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE, pp. 781-784 Vol.2, 1998.

[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.

[Rouas et al. 2003] Rouas J.-L., Farinas J., Pellegrino F. and André-Obrecht R., "Modeling Prosody for Language Identification on Read and Spontaneous Speech", Proc. IEEE ICASSP2003, vol 1, pp. 40-43, 2003.

[Schultz y Kirchoff 2006] Schultz, T., and K. Kirchhoff, Multi-lingual Speech Processing, Elsevier, Academic Press.

[Sugiyama 1991] Sugiyama M., "Automatic Language Recognition Using Acoustic Features". In Proceedings 1991 IEEE International Conference on Acoustics, Speech and Signal Processing, Toronto, Canada, May 1991.

[Torres-Carrasquillo 2002] Torres-Carrasquillo Pedro A., Singer Elliot, Kohler Mary A., Greene Richard J., Reynolds Douglas A., Deller Jr. J. R., "Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features", In ICSLP-2002, pp. 89-92, 2002.

[Torres-Carrasquillo et al, 2002] Torres-Carrasquillo, P., Reynolds, D., Deller, J. Jr.. Language identification using Gaussian mixture model tokenization. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Orlando, FL. on Spoken Language Processing. Denver, CO

[**Pham, 2007]** Tuan Van Pham, Wavelet Analysis For Robust Speech Processing and Applications, VDM Verlag Saarbrücken.

[Vargas-Martínez 2008] Vargas Martínez José Manuel, Un Método para la Identificación Automática de Lenguas Basado en la Transformada Wavelet, Tesis de Maestría, Instituto Tecnológico de Ciudad Madero, 2008.

[Witten y Frank 2005] Witten Ian H. and Frank Eibe, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

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

[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.

Anexo A Tablas de resultados de las pruebas

A.1. Justificación de NaiveBayes

A continuación se muestran todas las pruebas que se hicieron en los biclasificadores para justificar el uso de Naive Bayes sobre otros clasificadores.

A.1.1 Prueba de 3 segundos

Tabla A.1 – Prueba de 3 segundos con el clasificador IBK

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	68	83	75	78	70	74	79	74
Alemán		78	79	76	83	81	66	80
Español			82	76	75	82	78	80
Mandarín				83	80	81	82	84
Vietnamita					87	71	75	85
Japonés						80	76	78
Coreano							82	73
Tamil								73

Tabla A.2 – Prueba de 3 segundos con el clasificador J48

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	74	81	80	78	70	78	66	66
Alemán		78	74	70	76	77	78	72
Español			81	75	78	66	76	72
Mandarín				80	73	81	75	76
Vietnamita					78	69	79	87
Japonés						71	79	77
Coreano							74	72
Tamil								73

Tabla A.3 - Prueba de 3 segundos con el clasificador Lmt

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	78	79	78	85	80	80	85	76
Alemán		77	84	87	77	86	79	81
Español			83	82	78	82	78	76
Mandarín				86	78	82	79	74
Vietnamita					81	77	80	82
Japonés						83	83	83
Coreano							80	83
Tamil								81

Tabla A.4 - Prueba de 3 segundos con el clasificador Lwl

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	63	62	63	62	52	58	51	52
Alemán		60	55	60	59	62	61	58
Español			57	49	52	58	48	59
Mandarín				61	52	64	56	59
Vietnamita					62	53	54	55
Japonés						59	45	60
Coreano							59	56
Tamil								57

Tabla A.5 - Prueba de 3 segundos con el clasificador NaiveBayes

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	96	96	96	93	95	92	96	98
Alemán		89	94	92	90	95	93	95
Español			91	90	92	93	93	95
Mandarín				91	89	95	97	92
Vietnamita					92	88	92	95
Japonés						90	90	95
Coreano							94	93
Tamil								91

A.1.2. Prueba de 4 segundos

Tabla A.6 - Prueba de 4 segundos con el clasificador LBk

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	80	82	79	73	80	74	75	80
Alemán		77	78	70	89	81	75	76
Español			71	83	84	69	76	84
Mandarín				75	87	81	77	78
Vietnamita					80	64	75	81
Japonés						83	77	79
Coreano							79	79
Tamil								73

Tabla A.7 - Prueba de 4 segundos con el clasificador J48

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	77	73	78	74	65	73	70	79
Alemán		74	79	74	67	72	72	69
Español			73	80	84	69	76	77
Mandarín				72	82	75	77	80
Vietnamita					82	64	76	78
Japonés						77	70	82
Coreano							82	73
Tamil								72

Tabla A.8 - Prueba de 4 segundos con el clasificador Lmt

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	81	84	77	84	79	80	85	81
Alemán		85	83	80	80	87	75	79
Español			79	79	80	79	81	80
Mandarín				83	85	76	80	81
Vietnamita					83	78	79	85
Japonés						80	80	81
Coreano							84	81
Tamil								80

Tabla A.9 - Prueba de 4 segundos con el clasificador Lwl

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	64	66	52	62	52	56	61	63
Alemán		63	50	54	53	68	59	58
Español			58	48	46	62	56	56
Mandarín				52	60	63	64	53
Vietnamita					58	54	57	54
Japonés						57	52	61
Coreano							58	50
Tamil								58

Tabla A.10 - Prueba de 4 segundos con el clasificador Naivebaye

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	96	95	92	94	97	90	95	93
Alemán		96	93	92	95	96	93	91
Español			93	92	94	92	89	95
Mandarín				91	96	96	92	96
Vietnamita					93	90	94	95
Japonés						93	97	95
Coreano							91	92
Tamil								93

A.1.3. Prueba de 5 segundos

Tabla A.11 - Prueba de 5 segundos con el clasificador LBk

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	68	74	68	75	70	70	74	82
Alemán		80	75	81	79	82	73	79
Español			68	81	83	74	75	84
Mandarín				79	81	77	79	87
Vietnamita					78	79	75	78
Japonés						83	74	75
Coreano							86	79
Tamil								77

Tabla A.12 - Prueba de 5 segundos con el clasificador J48

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	78	71	68	76	79	75	79	66
Alemán		76	78	73	84	71	71	72
Español			84	71	75	65	73	70
Mandarín				70	73	62	76	74
Vietnamita					76	82	79	77
Japonés						76	69	80
Coreano							75	67
Tamil								76

Tabla A.13 - Prueba de 5 segundos con el clasificador Lmt

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	88	89	81	83	80	78	88	85
Alemán		77	82	86	82	78	79	86
Español			76	83	73	78	88	83
Mandarín				86	85	79	88	85
Vietnamita					85	81	74	78
Japonés						77	85	82
Coreano							79	83
Tamil								77

Tabla A.14 - Prueba de 5 segundos con el clasificador Lwl

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	66	57	57	52	60	58	68	61
Alemán		52	58	65	60	66	50	52
Español			58	58	54	58	51	54
Mandarín				54	59	52	64	59
Vietnamita					56	47	53	58
Japonés						63	60	59
Coreano							49	61
Tamil								60

Tabla A.15 - Prueba de 5 segundos con el clasificador NaiveBayes

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	94	96	90	96	95	88	95	100
Alemán		94	93	92	93	95	94	92
Español			93	92	93	95	95	93
Mandarín				89	96	91	93	95
Vietnamita					91	92	91	95
Japonés						95	92	95
Coreano							87	95
Tamil								93

A.1.4. Prueba de 10 segundos

Tabla A.16 - Prueba de 10 segundos con el clasificador LBk

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	66	82	65	72	74	61	77	78
Alemán		75	77	67	78	80	71	72
Español			79	82	83	75	77	73
Mandarín				72	72	76	82	68
Vietnamita					65	73	64	79
Japonés						74	73	68
Coreano							78	87
Tamil								75

Tabla A.17 - Prueba de 10 segundos con el clasificador J48

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	80	78	70	70	71	73	66	69
Alemán		69	73	77	69	73	69	78
Español			74	67	81	64	71	68
Mandarín				68	75	77	73	73
Vietnamita					72	64	74	75
Japonés						68	76	78
Coreano							68	70
Tamil								61

Tabla A.18 - Prueba de 10 segundos con el clasificador Lmt

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	80	77	78	78	84	67	75	78
Alemán		82	77	77	81	76	84	81
Español			80	80	84	76	76	80
Mandarín				80	82	76	77	78
Vietnamita					85	74	83	79
Japonés						82	82	72
Coreano							83	78
Tamil								79

Tabla A.19 - Prueba de 10 segundos con el clasificador Lwl

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	63	64	49	49	57	50	58	50
Alemán		47	54	54	65	54	53	54
Español			57	52	57	48	53	56
Mandarín				54	56	59	63	49
Vietnamita					60	48	56	54
Japonés						53	47	51
Coreano							56	52
Tamil								51

Tabla A.20 - Prueba de 10 segundos con el clasificador NaïveBayes

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	89	95	88	91	87	86	92	91
Alemán		91	90	94	87	96	91	91
Español			91	90	91	94	89	95
Mandarín				90	94	88	90	93
Vietnamita					92	95	89	95
Japonés						93	87	94
Coreano							85	92
Tamil								93

A.1.5. Prueba de 30 segundos

Tabla A.21 - Prueba de 30 segundos con el clasificador Ibk

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	53	54	56	72	64	68	69	75
Alemán		75	66	77	74	79	71	75
Español			74	79	81	82	76	72
Mandarín				83	67	70	66	74
Vietnamita					65	77	64	78
Japonés						81	68	69
Coreano							64	73
Tamil								68

Tabla A.22 - Prueba de 30 segundos con el clasificador J48

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	66	70	65	66	65	68	63	72
Alemán		73	74	74	70	83	67	69
Español			68	77	70	71	66	76
Mandarín				71	74	73	66	72
Vietnamita					73	81	59	71
Japonés						75	65	73
Coreano							67	75
Tamil								71

Tabla A.23 - Prueba de 30 segundos con el clasificador Lmt

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	71	74	75	79	81	74	84	77
Alemán		81	81	83	86	79	82	82
Español			83	83	78	78	80	81
Mandarín				79	85	84	76	79
Vietnamita					84	79	67	77
Japonés						82	71	70
Coreano							80	79
Tamil								79

Tabla A.24 - Prueba de 30 segundos con el clasificador Lwl

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	55	48	52	45	55	46	58	53
Alemán		46	53	48	50	58	51	61
Español			49	65	54	41	49	52
Mandarín				56	46	62	61	55
Vietnamita					52	55	56	53
Japonés						61	64	50
Coreano							45	55
Tamil								64

Tabla A.25 - Prueba de 30 segundos con el clasificador NaiveBayes

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	84	88	83	80	87	83	81	90
Alemán		94	93	90	93	93	84	88
Español			95	94	96	96	92	89
Mandarín				99	93	92	84	92
Vietnamita					92	92	76	89
Japonés						90	85	94
Coreano							86	91
Tamil								83

A.2 Resultados del uso de diferentes Wavelet.

Tabla A.26 – Prueba de 10 segundos para el wavelet Db2

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

Tabla A.27 - Prueba de 10 segundos para el wavelet Db4

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	93	94	91	95	89	94	93	96
Alemán		96	96	96	94	97	90	98
Español			97	94	92	95	92	98
Mandarín				94	96	98	85	96
Vietnamita					95	95	92	97
Japonés						98	90	92
Coreano							91	96
Tamil								95

Tabla A.28 - Prueba de 10 segundos para el wavelet Db6

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	95	98	91	94	93	93	95	97
Alemán		92	95	98	93	99	91	94
Español			94	97	93	93	94	93
Mandarín				95	97	96	88	89
Vietnamita					94	97	92	100
Japonés						94	89	95
Coreano							89	94
Tamil								94

Tabla A.29 - Prueba de 10 segundos para el wavelet Db10

	Alemán	Español	Mandarín	Vietnamita	Japonés	Coreano	Tamil	Farsi
Inglés	90	96	91	93	96	88	98	97
Alemán		93	92	98	94	93	95	96
Español			95	93	94	95	96	99
Mandarín				94	97	96	87	97
Vietnamita					96	97	89	95
Japonés						94	93	97
Coreano							85	96
Tamil								92

Anexo B Codigos

B.1 Código para calcular la transformada Wavelet Db mediante Praat.

Este código calcula la transformada Wavelet Dbn para archivos de audio. Para calcular las distintas Dbn se empleo el siguiente código, cambiando solamente en el apartado de `To Wavelet... n,` donde *n* es el coeficiente Dbn a calcular.

```
#Este programa se encarga de calcular la transformada wavelet db2 a
#señales de audio
#Código Jose Manuel Vargas, ITCM-DEPI 2008
#lee la lista de idiomas
Read Strings from raw text file... idiomas.txt
xx=1
id=1
final=0
#Manda a procesar 450 archivos de 50 en 50
while xx <450
#Toma el nombre del idioma dependiendo de "id"
select Strings idiomas
idioma$=Get string... 'id'
#se calcula el final del segmento
final=xx+49
#llama al procedimiento extraccion
call   extraction xx final 'idioma$'
xx=final +1
id=id+1
endwhile

#---------------------------------------------------------------

procedure extraction inicio fin nombre2$
#Código Jose Manuel Vargas, ITCM-DEPI

#Lee dos archivos con los nombres de los 450 archivos, la segunda lista
representa los nombres de los archivos, dentro de Praat, pues cambian
Read Strings from raw text file... archivos1.txt
Read Strings from raw text file... archivos2.txt
j=1
k=1
# este ciclo se encarga de mandar a cada nombre de audio al procedimiento
programage
```

```
for it from inicio to fin
 select Strings archivos1
 nombre$=Get string... 'it'
 select Strings archivos2
 nombre1$=Get string... 'it'
#Se manda a llamar a programege enviando le los nombres del archivo y el
idioma
 call programage 'nombre$' 'nombre1$' 'nombre2$'
# printline 'nombre$'
endfor
select Strings archivos1
Remove
select Strings archivos2
Remove
#select Strings idiomas
#Remove
endproc

procedure programage nombre$ nombre1$ nombre2$
#Este procedimiento segmenta la señal en segundos y aplica la
transformada Wavelet a cada uno de ellos
#Código Original: Ana Lilia Reyes Herrera-INAOE
#Modificación: José Manuel Vargas Martínez-ITCM-DEPI

#Carga una lista de números del 1 al 50, dado que no cuenta con funciones
de conversión de enteros a texto
Read Strings from raw text file... lista.txt
#se forma la ruta donde se guardaran los archivos ".wavelet"
ruta0$ = "OGI_Story_res\" + nombre2$ + "\"
#Ruta de la base de dato de archivos de audio
ruta$ = "OGI_Story_50s\" + nombre2$ + "\"
printline 'ruta$'
#Ruta completa del archivo de audio a segmentar
archivo$ = ruta$ + nombre$ + ".wav"
#Lectura del archivo
Read from file... 'archivo$'

#este ciclo se encarga de segmentar el archivo, calcular su transformada
wavelet, truncarla y posteriormente guardarla
#el ciclo depende de la duracion de la señal o el tiempo deseado
for i from 0 to 44
select Sound 'nombre1$'
#Se extrae un segmento de 1 segundo desde i hasta i+1
Edit
  editor Sound 'nombre1$'
  Select... 'i' 'i'+1
  Extract selection (preserve times)
Close
#Se aplica la transformada wavelet
 To Wavelet... 4
```

```
#Se trunca en una fraccion del 1%
 Truncate by fraction... 0.01
 select Strings lista
 num$=Get string... 'i'+1
 select Wavelet untitled_1
#se forma el nombre del archive que sera guardado
 archivot$=ruta0$+ nombre$+ num$ +".wavelet"
#Se guarda
 Write to text file... 'archivot$'
 select Sound untitled
 Remove
 select Wavelet untitled
 Remove
 select Wavelet untitled_1
 Remove
endfor
select Strings lista
Remove
select Sound 'nombre1$'
Remove
endproc
```

B.2 Codigo para convertir los resultados de la transformada Wavelet en medidas estadísticas.

```
function principalks
%Este programa se encarga de procesar los archivos de OGI_Story_res
%para lo cual los procesa por idioma y los convierte en archivos
%de conocimiento por idioma, que se guardan en la carpeta atributos
%Codigo:Jose Manuel Vargas Martinez, ITCM-DEPI 2008
%Modificación:César Medina Trejo, ITCM-DEPI 2010
clear all;
Numero_elem=50;%es el numero de archivos que se procesaran
Numero_segun=10;%es el numero de segundos que se procesaran
f=fopen('idiomas.txt','r');%se abre el archivo de nombres de idioma
lang=textscan(f,'%s',9);%se cargan los nombres
la=lang{:};%extraccion del contenido de la cell obsteniendo un cellstring
for i=1:9
    %se llama a atributos que se ebcarga de calcular los estadisticos
    atributosks(char(la(i)), Numero_elem, Numero_segun);
end

%--------------------------------------------------------------------------
```

```
function atributos(Name, Numero_elem, Numero_segun)
%Esta funcion se encarga de extraer las caracteristicas estadisticas
%de un señal que ha sido mapeada en wavelets
%Codigo:Jose Manuel Vargas Martinez, ITCM-DEPI 2008
%Modificación:César Medina Trejo, ITCM-DEPI 2010
clc
disp(Name);
N_coef=8192;%Numero de coeficientes
NameListFile=[Name '.txt'];%Archivo de la lengua que contiene la lista de
las señales
NameFileAtri=['Atributos\' Name '.txt'];%Archivo  donde se guardara los
atributos estadisticos
ruta=['Paso 1 - wavelets\OGI_Story_res\' Name '\'];%Ruta de los archivos
wavelet
sounds=fopen(NameListFile,'r');%Archivo que contiene la lista es abierto
fil=fopen(NameFileAtri,'w+');%Archivo que almacenara atributos es
arbierto
%Se convierte un cell a cellstring
sounds_lista = textscan(sounds, '%s',50);
sounds_lista=sounds_lista{:};
%----------------------------------------
m=[];
for i=1:Numero_elem %procesa cada archivo en la lista hasta Numero_elem
    nombrear = char(sounds_lista{i});
    for s=1:Numero_segun%procesa cada segundo del archivo actual hasta
Numero_segun
        rn=[ruta,nombrear,int2str(s),'.wavelet'];%Arma cada 1 de los
Numero_segun archivos .wavelet
        %-------------------------------------------------
        cd ..
        archivo=fopen(rn,'r');%archivo wavelet
        cd 'Paso 2 - intermedios'
        fseek(archivo,70,'bof');%se ubica en 70 caracteres debido al
formato de praat
        %y carga la lista de coeficientes en un vector cell
        %con una precicion de 20 numeos despues del punto
        C = textscan(archivo, '%*s %*s %*s %.20f',N_coef);
        c=C{:};%se convierte de cell a lista de enteros
        %-------------------------------------------------------
        %el segmento de codigo anterior delimitado, corresponde a
archivos wavelet hechos en praat, para los hechos en matlab, se resume en
una sola
        %linea:
        %c=load(rn);

        m=[m c];%se guarada c en una matriz
        fclose(archivo);
        %disp(i);
    end
    clear('C')
```

```
    clear('c');
    %La matriz contiene todos los Numero_segun cargados en columnas
    %pero como Matlab calcula estadisticos por columnas, hay que
transponer
    %la matriz
    m=m';
    %y claculamos las variables estaditicas
    media=mean(m);
    %varianza=var(m);
    desviacion=std(m);
    maximo=max(m);
    minimo=min(m);
    curtosis=kurtosis(m);
    obliquidade=skewness(m); %en frances, skewness se escribe
obliquidade, segun wikipedia..
   %formamos el vector de atributos para la señal
    atri=[media desviacion maximo minimo curtosis obliquidade];
    %y la guardamos en el archivo del idioma correspondiente con una
    %precision de 9 digitos despues del punto
    for j=1:N_coef*6
        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;
```

B.3 Código para crear los archivos biclasificadores a evaluar con WEKA (.arff) a partir de las medidas estadísticas obtenidas.

```
function combinar
%Este programa se encarga de formar archivos weka binarios
%para posteriormente clasificarlos en weka
%Codigo: Jose Manuel Vargas Martinez, ITCM-DEPI 2008
%Modificación:César Medina Trejo, ITCM-DEPI 2010
global C file idio ii jj tam1
%Nombres de idioma (para ya no cargarlos de archivo))
idiomas=char('Ingles','Aleman','Espanol','Mandarin','Vietnamita','Japones
','Koreano',...
    'Tamil','Farsi');
```

```
idio=cellstr(idiomas);
clear('idiomas');
%los nombres de los archivos arff tendran solo las iniciales
nom='IAEMVJKTF';
ext='.arff';%extencion de los archivos
contador=1;

%este ciclo construira 36 archivos arff
for i=1:8
    archivo1=['Paso  2  -  intermedios\Atributos\'  idio{i,:}  '.txt'];%se
forma el nombre del primer archivo de un idioma
    cd ..;
    A=load(archivo1);%se carga a la matriz A
    cd 'Paso 3 - archivos weka';
    tam1=size(A,1);%se obtiene su tamaño
    for j=i+1:9
        archivo2=['Paso 2 - intermedios\Atributos\' idio{j,:} '.txt'];%se
forma el segundo nombre del archivo
        cd ..
        B=load(archivo2);%Se carga
        cd 'Paso 3 - archivos weka';
        C=[A;B];%Se forma una matriz C con A y B
        clear('B')%se linera memoria
        %Se forma el nombre del archivo A-B.arff
        volcado=['weka\' nom(i) '-' nom(j) ext];
        file=fopen(volcado,'w+');%es abierto para escritura
        ii=i;jj=j;%se respaldan indices para usarlos
        cabecera();%Se imprime la cabecera del archivo
        %cabecera_c();
        %se guarda la matriz C depues de la cabecera
        fprintmat(C, file, idio, ii, jj, tam1);
        clear('C')
        fclose(file);
        disp(contador)
        contador=contador+1;
    end
    clear('A')
end

%-------------------------------------------------------------------------

function cabecera
global file idio ii jj
atributos=char('Media','Desviacion','Maximo','Minimo','Curtosis','Oblique
dad');
a=cellstr(atributos);
clear('atributos');
primero='@RELATION lenguajes';
%file=fopen('cabecera.txt','w+');
```

```
fprintf(file,'%s\n',primero);
for i=1:6
    for j=1:8192
        linea=['@ATTRIBUTE' ' ' a{i,:} int2str(j) ' ' 'REAL'];
        fprintf(file,'%s\n',linea);
    end
    fprintf(file,'\n');
end
linea=['@ATTRIBUTE class    {' idio{ii,:} ',' idio{jj,:} '}'];
fprintf(file,'%s\n',linea);
linea='@DATA';
fprintf(file,'%s\n\n',linea);
%fclose(file);

%-------------------------------------------------------------------------


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

B.4 Código para crear los archivos individuales a evaluar con WEKA (.arff) y la multiclasificación, a partir de las medidas estadísticas obtenidas.

```
function combinar_ind
%Este programa se encarga de formar archivos weka binarios
%para posteriormente clasificarlos en weka
%Codigo: Jose Manuel Vargas Martinez, ITCM-DEPI 2008
%Modificación:César Medina Trejo, ITCM-DEPI 2010
```

```
global idio nom ext
%Nombres de idioma (para ya no cargarlos de archivo))
idiomas=char('Ingles','Aleman','Espanol','Mandarin','Vietnamita','Japones
','Koreano',...
    'Tamil','Farsi');
idio=cellstr(idiomas);
clear('idiomas');
%los nombres de los archivos arff tendran solo las iniciales
nom='IAEMVJKTF';
ext='.arff';%extencion de los archivos
contador=1;
%este ciclo construira 36 archivos arff
for i=1:9
    archivo1=['Paso 2 - intermedios\Atributos\' idio{i,:} '.txt'];%se
forma el nombre del primer archivo de un idioma
    cd ..;
    A=load(archivo1);%se carga a la matriz A
    cd 'Paso 3 - archivos weka';
        %Se forma el nombre del archivo A-B.arff
        ii=i;%se respaldan indices para usarlos
        %cabecera_c();
        %se guarda la matriz C depues de la cabecera
        fprintmat_ind(A, idio, ii);
        clear('A')
        disp(contador)
        contador=contador+1;
    clear('A')
end
%--------------------------------------------------------------------------

function cabecera_ind(ii)
global file idio
atributos=char('Media','Desviacion','Maximo','Minimo','Curtosis','Obliqui
dade');
a=cellstr(atributos);
clear('atributos');
primero='@RELATION lenguajes';
%file=fopen('cabecera.txt','w+');
fprintf(file,'%s\n',primero);
for i=1:6
    for j=1:8192
        linea=['@ATTRIBUTE' ' ' a{i,:} int2str(j) ' ' 'REAL'];
        fprintf(file,'%s\n',linea);
    end
    fprintf(file,'\n');
end
linea=['@ATTRIBUTE class    {' idio{ii,:} '}'];
fprintf(file,'%s\n',linea);
linea='@DATA';
fprintf(file,'%s\n\n',linea);
```

```
%fclose(file);


%-------------------------------------------------------------------------


function fprintmat_ind(A, idio, ii)
global file nom ext
t=size(A);
for i=1:t(1)
    pos=ii;
        volcado=['weka\' nom(ii) '-' int2str(i)  ext];
        file=fopen(volcado,'w+');%es abierto para escritura
        disp (volcado);
        cabecera_ind(ii);%Se imprime la cabecera del archivo
    for j=1:t(2)
        fprintf(file, '%.9f',A(i,j));
        if j==t(2)
            fprintf(file, ', %s\n',idio{pos,:});
        else
            fprintf(file, ', ');
        end
    end
            fclose(file);
end
```

B.5 Código para evaluar los biclasificadores mediante NaiveBayes

```
//Código:César Medina Trejo, ITCM-DEPI 2010

package pruebaweka;
import java.io.*;
// FileReader
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.*;
// Random
import weka.core.*;
// Instances, Instance
import weka.filters.supervised.attribute.*;
// Discretize
import weka.classifiers.*;
import weka.attributeSelection.*;
// Classifier, Evaluation

public class Main_1 {

    static int classifier = 1, numfolds = 10;
    static int atributos=0;
    static double perct=0;
```

```
    static double pct = 50.0;

    public static void main(String[] args) throws Exception{
        //este programa se encarga de clasificar un conjunto de
instancias de lenguas mediante el uso de la tecnica Naive Bayes
        //Basado en Pruebaweka Codigo: Marco Antonio Aguirre Lam y Jose
Manuel Vargas Martinez ITCM-DEPI 2006

        String bufer;
        //Lista de archivos arff que seran evaluados
        FileInputStream fis=new
FileInputStream("UCI\\weka\\nombres.txt");
        //Tabla de resultados de las clasificaciones
        FileOutputStream fos=new
FileOutputStream("UCI\\weka\\tabla.txt");
        //Lectura y escritura de dichos archivos respectivamente
        BufferedReader d= new BufferedReader(new InputStreamReader(fis));
        BufferedWriter w=new BufferedWriter (new
OutputStreamWriter(fos));
        int contador=0;//Contador de los archivos arff
        do
        {
            bufer=d.readLine();//Se lee un nombre
            contador++;
            String linea="UCI\\weka\\",conv="",linean="";;
            linean=linea.concat(bufer);
            if(bufer!=null)//si no ha llegado al final
            {
                clasificar(linean);//ha clasificar el archivo arff
                //Desplegado en pantalla
                System.out.println(bufer);
                System.out.print(' ');
                System.out.print(perct);
                System.out.print(' ');
                System.out.print(atributos);
                //Grabado en archivo
                w.write(bufer);
                w.write(' ');
                w.write(conv.valueOf(perct));
                w.write(' ');
                w.write(conv.valueOf(atributos));
                w.newLine();
            }

        }while(contador<36);
        d.close();
        w.close();
        //clasificar();

    }
```

```
    public static void clasificar(String nombre_archivo_entrenamiento)
throws Exception{
        //esta funcion se encarga de cargar el archivo arff, filtrarlo y
clasificarlo
       Classifier C = seleccionarClassificador(classifier);//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));
       Evaluation Eval;//Objeto para evaluar las instancias filtradas
       //Se llama a validacionCruzada mandando el clasificador, las
instancias y el numero del folds
```

```
        Eval=ValidacionCruzada(C, filtradas, numfolds);//Devolviendo Eval
con la informacion
        perct=Eval.pctCorrect();//Porcentaje de clasificacion correcta
        atributos=filtradas.numAttributes();//obtencion del numero de
atributos des pues de filtrar
        System.out.println();
    }

    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();
        }
    }


    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;
    }

}
```

B.6 Código para evaluar las instancias individuales mediante multiclasificacion, por el método de eliminación

```
//Código:César Medina Trejo, ITCM-DEPI 2010

package pruebaweka;

import java.io.*;
// FileReader
import weka.classifiers.bayes.NaiveBayes;
import java.util.*;
import java.util.Queue;
// Random
import weka.core.*;
// Instances, Instance
// Discretize
import weka.classifiers.*;
import weka.attributeSelection.*;
// Classifier, Evaluation

public class Maineliminacion {
        //Basado en Pruebaweka Codigo: Marco Antonio Aguirre Lam y Jose
Manuel Vargas Martinez ITCM-DEPI 2006

    static int classifier = 1, numfolds = 10;
    static int atributos = 0;
    static double perct = 0;
    static double pct = 50.0;
    static double promedio=0;
    static String bufer;
    static String bayes = "weka.classifiers.bayes.NaiveBayes";
    static Queue fase = new LinkedList();
    static Queue fase_sig = new LinkedList();
    static char[] idiomas = {'A', 'E', 'F', 'I', 'J', 'K', 'M', 'T',
'V'};
    static int contador=0,y=0;
    public static void main(String[] args) throws Exception {
        //Aqui se realizara el sorteo para emparejar los idiomas para
seleccionar los biclasificadores
       for (y=0;y<9;y++)
       {
           System.out.println(idiomas[y]);
           promedio=0;
        for (int z=1;z<=50;z++)
        {
        Random num = new Random();
        boolean ban;
        int aux;
        char[] sorteo = {'A', 'E', 'F', 'I', 'J', 'K', 'M', 'T', 'V'};
        for (int cont = 0; cont < 9; cont++) {
```

```
            ban = true;
            while (ban) {
                aux = num.nextInt(9);
                if (sorteo[aux] != '0') {
                    fase.add(sorteo[aux]);
                 //   System.out.print(sorteo[aux]+" ");
                    sorteo[aux] = '0';
                    ban = false;
                }
            }
        }
System.out.println();

        //Metodo para crear el nombre de los biclasificadores, en caso de
que el nombre creado no se encuentre,
        //se invertira el nombre de los caracteres.

        int tamaño;
        clasifica(z);
        fase.add(fase_sig.remove());
        tamaño = fase.size();
       // System.out.println("tamaño " + tamaño);
        while (tamaño >= 2) {
            tamaño /= 2;
            for (int x = 0; x < tamaño; x++) {
                clasifica(z);//ha clasificar el archivo arff
            }
            fase=fase_sig;
            tamaño = fase.size();
            //System.out.println("tamaño " + tamaño);
        }
        if (fase.remove().toString().charAt(0)==idiomas[y])
            promedio++;
             // System.out.println("Promedio:" + promedio);
        }
        System.out.println("Promedio de "+idiomas[y]+ "es: " +
promedio*2);
        }
    }

    public static void clasifica(int z) throws Exception {
        char car1, car2;

        String archivo = "";
        String archivo2 = "";
        String linea = "UCI\\weka\\", conv = "", linean = "", instancia =
"";
        car1 = (Character) fase.remove();
        car2 = (Character) fase.remove();
        archivo = car1 + "-" + car2 + ".arff";
```

```
        archivo2 = car2 + "-" + car1 + ".arff";
        archivo = linea.concat(archivo);
        instancia = linea.concat(idiomas[y]+"-
"+String.valueOf(z)+".arff");
        //System.out.println(archivo);
        //System.out.println(instancia);
        try {
            clasificar(archivo, instancia);//ha clasificar el archivo
arff
        } catch (Exception e) {//si llego a esta excepcion,es debido a
que el nombre creado aleatoriamente para el
            //biclasificador esta invertido.
            // System.out.println(archivo+"excepcion");
            archivo2 = linea.concat(archivo2);
            clasificar(archivo2, instancia);//ha clasificar el archivo
arff
        }
    }


    public static void clasificar(String nombre_archivo_entrenamiento,
String instancia) throws Exception {
        //esta funcion se encarga de cargar el archivo arff, filtrarlo y
clasificarlo
        NaiveBayes C = new NaiveBayes();//Se elige el clasificador
        Instances entrenamiento =
cargarInstancias(nombre_archivo_entrenamiento);//Se cargan las instancias
        Instances prueba = cargarInstancias(instancia);
        boolean terminado, cargado;//Para verificacion
        Instances filtradas;//Donde se depositaran una vez filtradas las
instancias
        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


        //  String opciones[] = {"d",bufer.substring(0,bufer.length()-
5)+".model"};
        //  String opciones[] = {"-T",nombre_archivo_entrenamiento, "-
d",bufer.substring(0,bufer.length()-5)+".model"};
//C.setOptions(opciones);
        C.buildClassifier(entrenamiento);
        filtradas.randomize(new Random(1));
        Evaluation Eval = new Evaluation(entrenamiento);//Objeto para
evaluar las instancias
        // Eval.crossValidateModel(bayes, entrenamiento,
numfolds,opciones);
        //  Eval.evaluateModel(C, prueba);
        String resultado;
        char clase;
        Double valor;

        valor = Eval.evaluateModelOnce(C, prueba.firstInstance());
        resultado = Eval.toMatrixString();
        perct = Eval.pctCorrect();//Porcentaje de clasificacion correcta
        atributos = filtradas.numAttributes();//obtencion del numero de
atributos des pues de filtrar}
        StringTokenizer token = new StringTokenizer(resultado);
        //System.out.println(resultado);
        if (valor == 0.0) {
            for (int x = 1; x < 15; x++) {
                token.nextToken();
            }
            clase = token.nextToken().charAt(0);
           // System.out.println(clase);
            fase_sig.add(clase);

        } else {
            for (int x = 1; x < 21; x++) {
                token.nextToken();
            }
```

```
            clase = token.nextToken().charAt(0);
            //System.out.println(clase);
            fase_sig.add(clase);
        }
       // System.out.println();

    }

    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;
    }
}
```

B.7 Código para evaluar las instancias individuales mediante multiclasificacion, por el método de Round Robin con 2 grupos

```
//Código:César Medina Trejo, ITCM-DEPI 2010
package pruebaweka;
import java.io.*;
// FileReader
import weka.classifiers.bayes.NaiveBayes;
import java.util.*;
import java.util.Queue;
// Random
import weka.core.*;
// Instances, Instance
// Discretize
import weka.classifiers.*;
import weka.attributeSelection.*;
// Classifier, Evaluation

public class Main_dosgrupos {
        //Basado en Pruebaweka Codigo: Marco Antonio Aguirre Lam y Jose
Manuel Vargas Martinez ITCM-DEPI 2006



      static int classifier = 1, numfolds = 10;
      static int atributos = 0;
```

```
	static double perct = 0;
	static double pct = 50.0;
	static double promedio=0;
	static String bufer;
	static String bayes = "weka.classifiers.bayes.NaiveBayes";
	static Queue fase = new LinkedList();
	static Queue fase_sig = new LinkedList();
	static  char[] idiomas = {'A', 'E', 'F', 'I', 'J', 'K', 'M', 'T',
'V'};
	static int contador=0,y=0, mayorg1, mayorg2;
	static char ganadora, g1,g2;
	static char[] grupo1 = new char[5];
	static char[] ganadores = new char[2];
	static int[] grupo1i = new int[5];
	static int[] grupo2i =new int[4];
	static char[] grupo2 = new char[4];
	public static void main(String[] args) throws Exception {
		//Aqui se realizara el sorteo para emparejar los idiomas para
seleccionar los biclasificadores
		for (y=0;y<9;y++)
		{


			System.out.println(idiomas[y]);
			promedio=0;

			for (int z=1;z<=50;z++)
			{
				mayorg1=0;
				mayorg2=0;
				for (int a=0;a<grupo2i.length;a++)
				{
					grupo1i[a]=0;
					grupo2i[a]=0;
				}
				grupo1i[4]=0;


				Random num = new Random();
				boolean ban;
				int aux;
				char[] sorteo = {'A', 'E', 'F', 'I', 'J', 'K',
'M', 'T', 'V'};
				for (int cont = 0; cont < 9; cont++) {
					ban = true;
					while (ban) {
						aux = num.nextInt(9);
						if (sorteo[aux] != '0') {
							fase.add(sorteo[aux]);
```

```
                                        //System.out.print(sorteo[aux]+"
");
                                        sorteo[aux] = '0';
                                        ban = false;
                                  }
                            }
                      }

                      for (int a=0;a<=4;a++){

      grupo1[a]=fase.remove().toString().charAt(0);
                            //System.out.println(grupo1[a]);
                      }
                      for (int a=0;a<=3;a++)
                      {

      grupo2[a]=fase.remove().toString().charAt(0);
                            //System.out.println(grupo2[a]);
                      }

                      System.out.println(z);

                      //Metodo  para  crear  el  nombre  de  los
biclasificadores, en caso de que el nombre creado no se encuentre,
                      //se invertira el nombre de los caracteres.

                      for(int a=0;a<grupo1.length-1;a++)
                            for(int b=a+1;b<grupo1.length;b++)
                                  clasifica(z,a,b,grupo1);



                      for(int a=0;a<grupo2.length-1;a++)
                            for(int b=a+1;b<grupo2.length;b++)
                                  clasifica(z,a,b,grupo2);

                      for(int a=0;a<grupo1.length;a++)
                            if (grupo1i[a]>mayorg1)
                            {
                                        mayorg1=grupo1i[a];
                                        ganadores[0]=grupo1[a];
                            }

                      for(int a=0;a<grupo2.length;a++)
                                  if (grupo2i[a]>mayorg2)
                                  {
                                      mayorg2=grupo2i[a];
                                      ganadores[1]=grupo2[a];
                                  }
```

```
                              clasifica(z,0,1,ganadores);

                            //              System.out.println(ganadora+"
"+idiomas[y]);
                        if (ganadora==idiomas[y])
                              promedio++;
                        // System.out.println("Promedio:" + promedio);
                        //System.out.println("Promedio  de  "+idiomas[y]+
"es: " + promedio*2);
                  }
                  System.out.println("Promedio de "+idiomas[y]+ "es: " +
promedio*2);
            }
      }

      public static void clasifica(int z, int a, int b, char[]grupo)
throws Exception {
            char car1, car2;

            String archivo = "";
            String archivo2 = "";
            String linea = "UCI\\weka\\", conv = "", linean = "",
instancia = "";
            car1 = grupo[a];
            car2 = grupo[b];
            archivo = car1 + "-" + car2 + ".arff";
            archivo2 = car2 + "-" + car1 + ".arff";
            archivo = linea.concat(archivo);
            instancia            =            linea.concat(idiomas[y]+"-
"+String.valueOf(z)+".arff");
            //System.out.println(archivo);
            //System.out.println(instancia);
            try {
                  clasificar(archivo,   instancia);//ha   clasificar   el
archivo arff
            } catch (Exception e) {//si llego a esta excepcion,es debido
a que el nombre creado aleatoriamente para el
                  //biclasificador esta invertido.
                  // System.out.println(archivo+"excepcion");
                  archivo2 = linea.concat(archivo2);
                  clasificar(archivo2,   instancia);//ha   clasificar   el
archivo arff
            }
      }

      public static void clasificar(String nombre_archivo_entrenamiento,
String instancia) throws Exception {
            //esta funcion se encarga de cargar el archivo arff,
filtrarlo y clasificarlo
            NaiveBayes C = new NaiveBayes();//Se elige el clasificador
```

```
            Instances                    entrenamiento                    =
cargarInstancias(nombre_archivo_entrenamiento);//Se cargan las instancias
            Instances prueba = cargarInstancias(instancia);
            boolean terminado, cargado;//Para verificacion
            Instances filtradas;//Donde se depositaran una vez filtradas
las instancias
            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


            //                    String          opciones[]          =
{"d",bufer.substring(0,bufer.length()-5)+".model"};
            //   String  opciones[]  =  {"-T",nombre_archivo_entrenamiento,
"-d",bufer.substring(0,bufer.length()-5)+".model"};
            //C.setOptions(opciones);
            C.buildClassifier(entrenamiento);
            filtradas.randomize(new Random(1));
            Evaluation Eval = new Evaluation(entrenamiento);//Objeto para
evaluar las instancias
```

```
            //        Eval.crossValidateModel(bayes,        entrenamiento,
numfolds,opciones);
            //  Eval.evaluateModel(C, prueba);
            String resultado;
            char clase;
            Double valor;

            valor = Eval.evaluateModelOnce(C, prueba.firstInstance());
            resultado = Eval.toMatrixString();
            perct   =   Eval.pctCorrect();//Porcentaje   de   clasificacion
correcta
            atributos  =  filtradas.numAttributes();//obtencion  del  numero
de atributos des pues de filtrar}
            StringTokenizer token = new StringTokenizer(resultado);
            //System.out.println(resultado);
            if (valor == 0.0) {
                  for (int x = 1; x < 15; x++) {
                        token.nextToken();
                  }
                  clase = token.nextToken().charAt(0);
                  // System.out.println(clase);
                  puntaje(clase);

            } else {
                  for (int x = 1; x < 21; x++) {
                        token.nextToken();
                  }

                  clase = token.nextToken().charAt(0);
                  //System.out.println(clase);
                  puntaje (clase);
            }
            ganadora=clase;
            // System.out.println();
            //System.out.println(ganadora);

      }

      public static void puntaje (char clase)
      {
            for (int a=0;a<grupo1.length;a++)
                  if(grupo1[a]==clase)
                        {
                        grupo1i[a]+=3;
                            //    System.out.println(grupo1i[a]);
                        }

            for (int a=0;a<grupo2.length;a++)
                  if (grupo2[a]==clase)
                        {
```

```
                        grupo2i[a]+=3;
                              //   System.out.println(grupo2i[a]);
                        }
      }

      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;
      }
}
```

B.8 Código para evaluar las instancias individuales mediante multiclasificacion, por el método de Round Robin con 3 grupos

```
//Código:César Medina Trejo, ITCM-DEPI 2010
package pruebaweka;

import java.io.*;
// FileReader
import weka.classifiers.bayes.NaiveBayes;
import java.util.*;
import java.util.Queue;
// Random
import weka.core.*;
// Instances, Instance
// Discretize
import weka.classifiers.*;
import weka.attributeSelection.*;
// Classifier, Evaluation

public class Main3 {
        //Basado en Pruebaweka Codigo: Marco Antonio Aguirre Lam y Jose
Manuel Vargas Martinez ITCM-DEPI 2006


      static int classifier = 1, numfolds = 10;
      static int atributos = 0;
      static double perct = 0;
      static double pct = 50.0;
      static double promedio=0;
      static String bufer;
      static String bayes = "weka.classifiers.bayes.NaiveBayes";
      static Queue fase = new LinkedList();
      static Queue fase_sig = new LinkedList();
      static char[] idiomas = {'A', 'E', 'F', 'I', 'J', 'K', 'M', 'T',
'V'};
```

```
	static int contador=0,y=0, mayorg1, mayorg2,mayorg3,mayorfi;
	static char ganadora, g1,g2;
	static char[] ganadores = new char[3];
	  static char[] grupo1 = new char[3];
	  static char[] grupo2 = new char[3];
	static char[] grupo3 = new char[3];
	  static char[] grupof = new char[3];
	  static int[] grupo1i = new int[3];
	static int[] grupo2i =new int[3];
	  static int[] grupo3i = new int[3];
	  static int[] ganadoresi = new int[3];

	public static void main(String[] args) throws Exception {
		//Aqui se realizara el sorteo para emparejar los idiomas para seleccionar los biclasificadores
		for (y=0;y<9;y++)
		{


			System.out.println("idioma: "+idiomas[y]);
			promedio=0;

			for (int z=1;z<=50;z++)
			{
				mayorg1=0;
				mayorg2=0;
				mayorg3=0;
				mayorfi=0;
				for (int a=0;a<grupo1i.length;a++)
					{
						ganadoresi[a]=0;
						ganadores[a]=0;
					}



				Random num = new Random();
				boolean ban;
				int aux;
				char[] sorteo = {'A', 'E', 'F', 'I', 'J', 'K', 'M', 'T', 'V'};
				for (int cont = 0; cont < 9; cont++) {
				  ban = true;
				  while (ban) {
					aux = num.nextInt(9);
					if (sorteo[aux] != '0') {
					  fase.add(sorteo[aux]);
					  //System.out.print(sorteo[aux]+" ");
					  sorteo[aux] = '0';
					  ban = false;
					}
				  }
				}

				for (int a=0;a<=2;a++){
```

```
		grupo1[a]=fase.remove().toString().charAt(0);
					//System.out.println(grupo1[a]);
				}
				for (int a=0;a<=2;a++)
				{

		grupo2[a]=fase.remove().toString().charAt(0);
					//System.out.println(grupo2[a]);
				}

					for (int a=0;a<=2;a++)
				{

		grupo3[a]=fase.remove().toString().charAt(0);
					//System.out.println(grupo3[a]);
				}

				System.out.println(z);

				//Metodo para crear el nombre de los
biclasificadores, en caso de que el nombre creado no se encuentre,
				//se invertira el nombre de los caracteres.

				for(int a=0;a<grupo1.length-1;a++)
					for(int b=a+1;b<grupo1.length;b++)
						clasifica(z,a,b,grupo1);

					//System.out.println();


				for(int a=0;a<grupo2.length-1;a++)
					for(int b=a+1;b<grupo2.length;b++)
						clasifica(z,a,b,grupo2);

					//System.out.println();

					for(int a=0;a<grupo3.length-1;a++)
					for(int b=a+1;b<grupo3.length;b++)
						clasifica(z,a,b,grupo3);

						//System.out.println();

				for(int a=0;a<grupo1.length;a++)
					if (grupo1i[a]>mayorg1)
					{
							mayorg1=grupo1i[a];
							ganadores[0]=grupo1[a];
					}

				for(int a=0;a<grupo2.length;a++)
						if (grupo2i[a]>mayorg2)
						{
							mayorg2=grupo2i[a];
							ganadores[1]=grupo2[a];
						}
```

```
                        for(int a=0;a<grupo3.length;a++)
                                if (grupo3i[a]>mayorg3)
                                {
                                    mayorg3=grupo3i[a];
                                    ganadores[2]=grupo3[a];
                                }

                    for (int a=0;a<grupo1i.length;a++)
                    {
                          grupo1i[a]=0;
                          grupo2i[a]=0;
                                    grupo3i[a]=0;
                                    grupo1[a]='X';
                                    grupo2[a]='X';
                                    grupo3[a]='X';
                           }

                            for(int a=0;a<ganadores.length-1;a++)
                          for(int b=a+1;b<ganadores.length;b++)
                              clasifica(z,a,b,ganadores);

                                    // System.out.println();


                            for(int a=0;a<ganadores.length;a++)
                                if (ganadoresi[a]>mayorfi)
                                {
                                    mayorfi=ganadoresi[a];
                                    ganadora=ganadores[a];
                                }



                                //System.out.println("ganadora "+
ganadora+"  idioma"+idiomas[y]);
                    if (ganadora==idiomas[y])
                          promedio++;

                                // System.out.println();
                    // System.out.println("Promedio:" + promedio);
                    //System.out.println("Promedio de "+idiomas[y]+
"es: " + promedio*2);
                }
                System.out.println("Promedio de "+idiomas[y]+ "es: " +
promedio*2);
            }
      }

      public static void clasifica(int z, int a, int b, char[]grupo)
throws Exception {
            char car1, car2;

            String archivo = "";
            String archivo2 = "";
            String linea = "UCI\\weka\\", conv = "", linean = "",
instancia = "";
```

```
            car1 = grupo[a];
            car2 = grupo[b];
            archivo = car1 + "-" + car2 + ".arff";
            archivo2 = car2 + "-" + car1 + ".arff";
            archivo = linea.concat(archivo);
            instancia = linea.concat(idiomas[y]+"-
"+String.valueOf(z)+".arff");
            //System.out.println("archivo "+ archivo);
            //System.out.println("instancia "+instancia);
            try {
                  clasificar(archivo, instancia);//ha clasificar el
archivo arff
            } catch (Exception e) {//si llego a esta excepcion,es debido
a que el nombre creado aleatoriamente para el
                  //biclasificador esta invertido.
                  // System.out.println(archivo+"excepcion");
                  archivo2 = linea.concat(archivo2);
                  clasificar(archivo2, instancia);//ha clasificar el
archivo arff
            }
      }

      public static void clasificar(String nombre_archivo_entrenamiento,
String instancia) throws Exception {
            //esta funcion se encarga de cargar el archivo arff,
filtrarlo y clasificarlo
            NaiveBayes C = new NaiveBayes();//Se elige el clasificador
            Instances entrenamiento =
cargarInstancias(nombre_archivo_entrenamiento);//Se cargan las instancias
            Instances prueba = cargarInstancias(instancia);
            boolean terminado, cargado;//Para verificacion
            Instances filtradas;//Donde se depositaran una vez filtradas
las instancias
            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


            //  String opciones[] =
{"d",bufer.substring(0,bufer.length()-5)+".model"};
            //  String opciones[] = {"-T",nombre_archivo_entrenamiento,
"-d",bufer.substring(0,bufer.length()-5)+".model"};
            //C.setOptions(opciones);
            C.buildClassifier(entrenamiento);
            filtradas.randomize(new Random(1));
            Evaluation Eval = new Evaluation(entrenamiento);//Objeto para
evaluar las instancias
            // Eval.crossValidateModel(bayes, entrenamiento,
numfolds,opciones);
            //  Eval.evaluateModel(C, prueba);
            String resultado;
            char clase;
            Double valor;

            valor = Eval.evaluateModelOnce(C, prueba.firstInstance());
            resultado = Eval.toMatrixString();
            perct = Eval.pctCorrect();//Porcentaje de clasificacion
correcta
            atributos = filtradas.numAttributes();//obtencion del numero
de atributos des pues de filtrar}
            StringTokenizer token = new StringTokenizer(resultado);
            //System.out.println(resultado);
            if (valor == 0.0) {
                  for (int x = 1; x < 15; x++) {
                        token.nextToken();
                  }
                  clase = token.nextToken().charAt(0);
                   //System.out.println("clase ganadora "+ clase);
                  puntaje(clase);

            } else {
                  for (int x = 1; x < 21; x++) {
                        token.nextToken();
                  }

                  clase = token.nextToken().charAt(0);
                  //System.out.println("clase ganadora "+ clase);
                  puntaje (clase);
            }
            ganadora=clase;
            // System.out.println();
            //System.out.println(ganadora);

      }
```

```
	public static void puntaje (char clase)
	{
		for (int a=0;a<grupo1.length;a++)
			if(grupo1[a]==clase)
				{
				grupo1i[a]+=3;
					//System.out.println("puntuacion"
+grupo1i[a]);
				}

		for (int a=0;a<grupo2.length;a++)
			if (grupo2[a]==clase)
				{
				grupo2i[a]+=3;
					//System.out.println("puntuacion"
+grupo2i[a]);
				}

			for (int a=0;a<grupo3.length;a++)
			if (grupo3[a]==clase)
				{
				grupo3i[a]+=3;
					//System.out.println("puntuacion"
+grupo3i[a]);
				}


			for (int a=0;a<ganadores.length;a++)
			if (ganadores[a]==clase)
				{
				ganadoresi[a]+=3;
					//System.out.println("puntuacion"
+ganadoresi[a]);
				}
	}

	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;
	}
}
```