Universidad Michoacana de San Nicolás de Hidalgo
División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica

BÚSQUEDA DE ELOCUCIONES DE PALABRAS O FRASES EN ARCHIVOS DE AUDIO

TESIS

Que para obtener el grado de
MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta
José Martín Ruiz Pérez

Dr. José Antonio Camarena Ibarrola

Director de Tesis

Morelia, Michoacán, México Abril 2016

BÚSQUEDA DE ELOCUCIONES DE PALABRAS O FRACES EN ARCHIVOS DE AUDIO

Los Miembros del Jurado de Examen de Grado aprueban
la **Tesis de Maestría en Ciencias en Ingeniería Eléctrica** de ***José Martín Ruíz Pérez***

Dr. Félix Calderón Solorio
Presidente del Jurado

Dr. José Antonio Camarena Ibarrola
Director de Tesis

Dr. Leonardo Romero Muñoz
Vocal

Dr. Juan José Flores Romero
Vocal

Dr. Héctor Tejeda Villela
Revisor Externo (FISMAT)

Dr. Félix Calderón Solorio
Jefe de la División de Estudios de Posgrado
de la Facultad de Ingeniería Eléctrica. UMSNH
(Por reconocimiento de firmas).

UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
Septiembre 2015

A toda mi familia, a mi novia miry y mis amigos, les agradezco todo el apoyo incondicional que me han brindado todo este tiempo.

Agradezco también a mi asesor y a mis profesores por enriquecer de forma importante este trabajo. A ellos y a la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, en general a la Universidad Michoacana de San Nicolas de Hidalgo, les ofrezco mi gratitud por el apoyo durante este proceso de preparación.

Lista de Publicaciones

Índices de Proximidad en el Reconocimiento de Voz
Ruiz Pérez, J.M. y Camarena-Ibarrola, A.
2013 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)

Query by Example Keyword Spotting in Streams of Audio
Camarena-Ibarrola, A. y Ruiz-Pérez, M.
2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)

Resumen

La búsqueda por contenido en archivos de audio consiste en localizar el o los archivos de audio de una colección de archivos en los que ocurre una elocución de una palabra o frase, además del instante preciso de dicha ocurrencia dentro del archivo de audio.

Las implementación de un sistema que lleve a cabo búsquedas por contenido en archivos de audio tiene mucha importancia debido a la gran cantidad de información multimedia que se genera en la actualidad. Este tipo de sistemas puede ser utilizado en cualquier sistema de cómputo que tenga los medios para almacenar, capturar o reproducir audio, e.g., un dispositivo móvil, un sistema empotrado o una computadora personal. El enfoque con el que se abordo este problema permite realizar el monitoreo de conversaciones.

En el estado del arte un sistema como el que se plantea tiene como base un LVCSRS (Large Vocabulary Continuous Speech Recognition System), por lo tanto el sistema depende de la precisión del LVCSRS al momento de realizar el reconocimiento de voz. En este trabajo se implementó un sistema que es dependiente del hablante, este utiliza un enfoque que no había sido probado en la búsqueda por contenido en grabaciones de voz. Dicho enfoque consiste en llevar el problema del reconocimiento de voz al uso de técnicas de alineamiento aproximado de cadenas. Donde el problema es solucionado sin la necesidad de hacer una transcripción de modelos de fonemas concatenados a texto como lo hace un LVCSRS. En cambio el esquema planteado realiza un alineamiento entre símbolos o vectores de características de la consulta y del archivo en el que se está buscando.

El sistema desarrollado en todas sus versiones (utilizando las cinco técnicas de alineamiento incorporadas) es capaz de localizar el instante donde ocurre la palabra o la frase utilizada como consulta y con resultados bastante gratos. Si el sistema utiliza LCS como técnica de alineamiento se obtiene una tasa de aciertos de 0.9723. Con la distancia de edición que es una generalización de LCS, el sistema tiene una tasa de aciertos de 0.9691. Las técnicas de alineamiento como la distancia de edición, LCS y twLCS dependen de un valor llamado criterio de cercanía. El DTW y una modificación de la distancia de edición que llamamos Levenshtein, son técnicas de alineamiento que no dependen de dicho criterio de cercanía. La ventaja de no depender de un valor que se debe definir previamente no se ve reflejado en el desempeño debido a que la tasa de aciertos para DTW y Levenshtein son de 0.8431 y 0.7886 respectivamente.

Palabras clave: ASM, DTW, MFCC, ROC, AUC.

Abstract

Content-based search in audio files consists in locating audio files from a collection of files in which an utterance of a word or phrase occurs and, the precise instant of that occurrence in the audio file.

The implementation of a system that performs content-based search in audio files has a lot of importance due to the great amount of multimedia data that is generated nowadays. This kind of systems can be used in any computer system that has the means to store, record or play audio, e.g., a mobile device, an embedded system or a personal computer. The approach on which this problem was focused allows us to perform conversations monitoring.

In the state-of-the-art, a system like the one mentioned above has as a base a LVCSRS (*Large Vocabulary Continuous Speech Recognition System*) so, the system depends on the accuracy of the LVCSRS when it performs the speech recognition task. In this work a speaker-dependent system was developed, it uses an approach that had not been tested in content-based search of speech recordings. This approach consists on taking the problem from the speech recognition to the use of approximate string matching. Where the problem is solved without the need to transcript the concatenated phoneme models to a text like in the LVCSRS approach. Instead, the scheme we present carries out an alignment between symbols or characteristic vectors of the query and the file in which we are searching.

The system developed in all its versions (using the five alignment technics) is capable to locate the instant in which the word or the phrase used as query takes place; its performance is fairly pleasant. If the system uses LCS as alignment technic, it obtains a success rate of 0.9723. With the edit distance that is a generalization of LCS, the system has a success rate of 0.9691. Alignment technics like the edit distance, LCS and twLCS depend on a value called proximity criterion. DTW and a modification of the edit distance that we call Levenshtein, are alignment technics that do not depend on such proximity criterion. The advantage of not depending on a value that must be previously defined seems not to take effect on the performance because, the success rate obtained when the system use DTW and Levenshtein are of 0.8431 and 0.7886 respectively.

Contenido

Dedicatoria III
Lista de Publicaciones V
Resumen VII
Abstract IX
Contenido XI
Lista de Figuras XIII
Lista de Tablas XV
Lista de Símbolos XVII
Lista de Algoritmos XIX
Glosario XXII

1. Introducción 1
   1.1. Planteamiento del Problema 1
   1.2. Motivación 3
   1.3. Objetivos de la Tesis 5
      1.3.1. Objetivo general 5
      1.3.2. Objetivos particulares 5
   1.4. Descripción de Capítulos 5

2. Estado del Arte 7
   2.1. Antecedentes 7
   2.2. Marco Teórico 13
      2.2.1. Procesamiento digital de la señal de voz 13
      2.2.2. Técnicas de alineamiento 24
   2.3. Conclusiones del Capítulo 41

3. Diseño e Implementación 43
   3.1. Diseño del sistema 43
   3.2. Implementación del sistema 50
      3.2.1. Obtención de los MFCC 50
      3.2.2. Obtención de los coeficientes Delta y Delta-Delta 52
      3.2.3. Proceso de búsqueda en audio 55
   3.3. Conclusiones del Capítulo 59

4. Experimentos y Resultados 61
4.1. Entorno de experimentación 61
4.2. Grabación y extracción de características 62
4.3. Experimentos de búsqueda 63
4.3.1. Análisis del desempeño de las técnicas de alineamiento mediante Curvas ROC 68
4.4. Evaluación del sistema con diferentes hablantes 70
4.4.1. Resultados de la evaluación 70
4.5. Tiempos promedio de ejecución 71
4.6. Conclusiones del Capítulo 75

5. Conclusiones y Trabajos Futuros 77
5.1. Conclusiones 77
5.1.1. Conclusiones particulares 77
5.2. Trabajos Futuros 78

Referencias 81

Lista de Figuras

2.1. Flujo del procesamiento digital de la señal de voz. 14
2.2. Ventaneo de una señal de voz. 15
2.3. Representación canónica de un filtro homomórfico. 19
2.4. Representación canónica de un filtro homomórfico para señales convolucionadas. 19
2.5. Representación canónica de un filtro homomórfico para señales convolucionadas. 20
2.6. Ventaneo de una señal de voz. 22
2.7. Banco de filtros triangulares de Mel. 23
2.8. Escala de Mel. 24

3.1. Inicio del proceso de búsqueda de una elocución en una base de datos de grabaciones de audio. 44
3.2. Configuración inicial del alineamiento de dos matrices de características. . . 46
3.3. Distancias obtenidas a lo largo del alineamiento de una palabra o frase con un flujo de audio. 48
3.3. Distancias obtenidas a lo largo del alineamiento de una palabra o frase con un flujo de audio. (*Continuación*) 49
3.4. Séptimo filtro de Mel del banco de filtros de la Figura 2.7. 51

4.1. Curvas ROC para diferentes valores de cercanía. 64
4.1. Curvas ROC para diferentes valores de cercanía. (*Continuación*) 65
4.1. Curvas ROC para diferentes valores de cercanía. (*Continuación*) 66
4.1. Curvas ROC para diferentes valores de cercanía. (*Continuación*) 67
4.1. Curvas ROC para diferentes valores de cercanía. (*Continuación*) 68

Lista de Tablas

2.1. Ejemplo del cálculo de la distancia de edición entre las cadenas r =“prosa” y s =“programa”. 25
2.2. Ejemplo del cálculo de LCS entre las cadenas r =“prosa” y s =“programa”. 26
2.3. Ejemplo del cálculo de twLCS entre las cadenas r =“prosa” y s =“programa”. 27
2.4. Cálculo de la distancia entre las cadenas r_1 =“4567” y s_1 =“41516171”. . . 28
2.5. Cálculo de la distancia entre las cadenas r_2 =“4567” y s_2 =“44556677”. . . 30
2.6. Cálculo de la distancia entre las cadenas r_3 =“4567” y s_3 =“4455661111177”. 31
2.7. Ejemplo de la búsqueda del patrón q =“con” en el texto o =“con el conocer”. 33
2.8. Ejemplo de la búsqueda del patrón q =“cccooooonn” en el texto o =“con el conocer”. 35
2.8. Ejemplo de la búsqueda del patrón q =“cccooooonn” en el texto o =“con el conocer”. (*Continuación*) 36
2.9. Condiciones de frontera y restricciones locales. 37
2.10. Cálculo de la distancia entre las secuencias R =4567 y S =44556677. 37
2.11. Ejemplo de la búsqueda del patrón q =“cccooooonn” en el texto o =“con el conocer”. 38
2.12. Cálculo de la distancia entre las cadenas R =4567 y S =44556677. 39
2.13. Ejemplo de la búsqueda del patrón q =“cccooooonn” en el texto o =“con el conocer”. 40

3.1. Equivalencia entre hertz y mels. 52

4.1. Contabilización de FP, TN, TP, FN para cada valor del umbral. 71
4.1. Contabilización de FP, TN, TP, FN para cada valor del umbral. (*Continuación*) 72
4.2. Tiempos promedio de búsqueda. TA: Técnica de Alineamiento; CC: Criterio de Cercanía. 73

Lista de Símbolos

α	Coeficiente del filtro de preénfasis
$x[n]$	Señal discreta de entrada en el dominio del tiempo
$y[n]$	Señal discreta de salida en el dominio del tiempo
n	Índice que denota la posición de la n-ésima muestra de una señal discreta
w	Ventana tomada de una señal de audio
t	Índice que denota la variable de tiempo
v_i	Conjunto de valores que puede tomar una señal de audio
p_i	Probabilidad de que ocurra el valor v_i
$I(v_i)$	Contenido de información del valor v_i
$H = E[I]$	Entropía de la señal o valor esperado de $I(v_i)$
σ^2	Varianza muestral
$\overline{x}$	Media muestral
NB	Número de *bins* de un histograma
$c[n]$	Cepstrum
$\hat{x}[n]$	Cepstrum complejo
$X[k]$	Señal discreta en el dominio de la frecuencia
L	Transformación Lineal
$A_\circ, A_\circ^{-1}$	Sistema característico y su inverso
D_*, D_*^{-1}	Sistema característico que realiza la deconvolución de dos señales y su inverso
T	Transformación Lineal
$h[n]$	Señal de Hamming discreta en el dominio del tiempo
$X_s[k]$	Señal discreta en el dominio de la frecuencia producto del ventaneo de $x[n]$
$c_s[n]$	Cepstrum producto del ventaneo de $x[n]$
$\hat{x}_s[n]$	Cepstrum complejo producto del ventaneo de $x[n]$
N	Tamaño de ventana, también llamada marco
$A_{i,j}$	para $i = 1, 2, \ldots, \lvert r\rvert$ y $j = 1, 2, \ldots, \lvert s\rvert$ Matriz de alineamiento
r_i	Representación de una cadena de texto
s_j	Representación de una cadena de texto

C_i	Columna de alineamiento
$C_i^{'}$	Columna de alineamiento
q_i	Representación de un patrón de texto
o_j	Representación de un texto
$D_{i,j}$	para $i = 0, 1, \ldots, \|R\|$ y $j = 0, 1, \ldots, \|S\|$ Matriz de alineamiento
R	Secuencia de vectores de características
S	Secuencia de vectores de características
$d_{i,j}$	Distancia entre el i-ésimo vector de R y el j-ésimo de S
$MC_{t,m}$	Matriz de características de una grabación de consulta
$MC_{t,m}^{'}$	Matriz de características de una grabación de referencia
T	Número de marcos
M	Número de filtros de Mel
D	Dimensión de un vector
$\mathbb{R}^d$	Espacio vectorial de dimensión D en los reales
d	Distancia coseno
$f[k]$	Frecuencia correspondiente a la muestra k
f_s	Frecuencia de muestreo
$f_c[m]$	Frecuencia central del m-ésimo filtro de Mel
$H[t, m]$	Kernel del banco de filtros de Mel
m	Índice que denota al m-ésimo filtro de Mel
$X^{'}[m]$	Espectro escalado en frecuencia y magnitud
ϕ	Valor aproximado de una frecuencia en la escala de Mel
$c[l]$	Coeficientes cepstrales de Mel obtenidos de la Transformada Discreta Coseno

Lista de Algoritmos

1. Extrae la Matriz de Características. 54
2. Regresa una lista con las posiciones donde hubo una ocurrencia. 56
3. Procedimiento para el cálculo de la distancia de edición entre MC y el vector de MC' en un determinado instante. 57
4. Procedimiento para el cálculo de LCS entre MC y el vector de MC' en un determinado instante. 57
5. Procedimiento para el cálculo de twLCS entre MC y el vector de MC' en un determinado instante. 58
6. Procedimiento para el cálculo de Levenshtein entre MC y el vector de MC' en un determinado instante, usando $2d$ como costo de la operación de sustitución. 58
7. Procedimiento para el cálculo de DTW entre MC y el vector de MC' en un determinado instante. 58

Glosario

AFP *Audio-Fingerprint (Firma de audio)*

ASM *Approximate String Matching (Emparejamiento de cadenas aproximado)*

AUC *Area Under the ROC Curve*

DCT *Discrete Cosine Transform (Transformada discreta coseno)*

DTW *Dynamic Time Warping (Doblado dinámico en tiempo)*

FFT *Fast Fourier Transform (Transformada rápida de Fourier)*

FIR *Finite Impulse Response (Respuesta finita al impulso)*

GMM *Gaussian Mixture Models (Modelos de mezclas de Gaussianas)*

HMM *Hidden Markov Models (Modelos Ocultos de Markov)*

IR *Information Retrieval (Recuperación de información)*

KWS *Keyword Spotting (Detección de palabras clave)*

LCS *Longest Common Subsequence (Subsecuencia común más larga)*

LSTM *Long short-term memory*

LVCSRS *Large Vocabulary Continuous Speech Recognition System (Sistema de reconocimiento continuo de voz de vocabulario grande)*

MFCC *Mel-Frequency Cepstral Coefficients (Coeficientes Cepstrales de Mel)*

PDF *Probability Density Function (Función de densidad de probabilidad)*

PDS *Procesamiento Digital de Señales*

RNN *Recurrent Neural Network (Red Neuronal Recurrente)*

ROC *Receiver Operating Characteristic*

SDR *Spoken Document Retrieval (Recuperación de documentos de voz)*

STD *Spoken Term Detection (Detección de términos de voz)*

SVM *Support Vector Machine (Maquina de soporte vectorial)*

twLCS *time warped Longest Common Subsequence (Subsecuencia común más larga con doblado en el tiempo)*

VQ *Vector Quantization (Cuantización vectorial)*

Capítulo 1

Introducción

El procesamiento digital de señales permite la manipulación en software de señales multimedia como son imágenes, audio, video, etc. El tratamiento de estas señales de la mano de otras áreas de las ciencias de la computación como el reconocimiento de patrones o el uso de índices de proximidad permiten implementar aplicaciones que se pueden integrar en software para edición y creación de imágenes (Inkscape), audio y video (Audacity, iMovie), diagnóstico de enfermedades (Symcat), dictado de comandos por voz en dispositivos móviles (Siri), etc.

Como parte de esta gama de aplicaciones que se mencionó anteriormente está la necesidad de localizar palabras o secuencias de palabras en una grabación de voz (la voz es tratada de manera similar a la música), ya sea que tengan una validez sintáctica y semántica o no.

En este capítulo se presenta el problema de la detección de una elocución[1] en una grabación de voz, también se hace mención del contexto que lo acompaña y la solución propuesta a dicho problema.

1.1. Planteamiento del Problema

La búsqueda de palabras o frases en una grabación de voz pertenece a la gama de aplicaciones que se derivan del reconocimiento de voz. Las grabaciones de voz continua no

[1] El acto de pronunciar; articular; hablar

contienen elocuciones separadas por silencios, por tanto no es posible utilizar un esquema de reconocimiento de palabras aisladas [Rabiner, 1989].

Nuestra solución consiste en la implementación de algoritmos de alineamiento utilizados en el área del emparejamiento aproximado de cadenas (ASM del Inglés *Approximate String Matching*), como son la distancia de Levenshtein también llamada distancia de edición y la subsecuencia común más larga (LCS del Inglés *Longest Common Subsequence*), que tienen como objetivo localizar un determinado patrón (cadena de texto) en un texto sin importar en que posición del texto se encuentre dicho patrón. También se aplicaron variantes de estos algoritmos que son usados en la recuperación de música (*Music Retrieval*), tales como la subsecuencia común más larga con doblado en el tiempo (twLCS del Inglés *Time Warped Longest Common Subsequence*) [Yang, 2002]. Por último, ya que se trabajó con grabaciones de voz, se utilizará el doblado dinámico en tiempo (DTW del Inglés *Dynamic Time Warping*) [Sakoe y Chiba, 1978] debido a su amplio uso en el reconocimiento de voz.

Un sistema de búsqueda de palabras en archivos de audio captura una elocución y realiza un análisis espectral de la señal para extraer una matriz de características. Previamente este mismo proceso es aplicado al archivo de audio en el que se va a buscar. Cada vector de características es comparado con cada vector de la elocución de consulta. Al final se observan las distancias obtenidas a lo largo del alineamiento y diremos que la elocución ocurrió en el marco de tiempo donde la distancia tenga un valor menor a un umbral definido previamente.

Si hacemos una analogía entre los términos usados en el ASM y los elementos de nuestro marco de trabajo descritos en el párrafo anterior, podemos decir que la elocución consulta es vista como el patrón buscado y cada grabación de la base de datos como un documento de texto, a la que llamaremos cadena de eventos acústicos [Camarena-Ibarrola y Chávez, 2006], [Camarena-Ibarrola et al., 2009]. Este enfoque permite localizar en que instante ocurrió la elocución consulta dentro de la grabación sin llegar a realizar un reconocimiento continuo de la voz.

Al igual que en el reconocimiento de voz, el tipo de problema que se acaba de plantear tiene dos enfoques a resolver. El primero consiste en el desarrollo de un método que dependa del hablante y el segundo que dicho sistema permita a un usuario realizar

búsquedas en grabaciones que no contengan necesariamente su voz (sistema independiente del hablante). Para el caso del método desarrollado en este trabajo tanto las grabaciones en donde se realizará la búsqueda como las consultas, necesariamente tienen que ser grabaciones del mismo usuario.

1.2. Motivación

En la actualidad las bases de datos multimedia forman parte importante del cúmulo de información generado por el uso cada vez más cotidiano de sistemas computacionales. Por tanto cualquier trabajo en esa línea de investigación es justificada por el impacto positivo que brinda a la sociedad. Para dar un ejemplo de este tipo de aplicaciones pensemos en un buscador de archivos, este tipo de aplicación es muy común en cualquier sistema operativo, ya sea un sistema operativo para dispositivos móviles o para una computadora personal. Imaginemos que dicha aplicación no estuviera limitada a localizar archivos con contenido de texto usando cadenas de texto para realizar la consulta. En cambio que esta aplicación se extendiera de tal manera que podamos realizar búsquedas en archivos de audio usando nuestra propia voz como ejemplo para realizar la búsqueda. Definitivamente una aplicación de este tipo podría ser muy útil para un sistema de información como lo es un sistema operativo.

En lo que respecta al sector privado, entidades de la industria de telecomunicación específicamente la de radio y televisión son las principales generadoras de contenido de audio. Un sistema de búsqueda en audio para estas entidades sería una herramienta útil. En general es muy común la localización previa de ciertas partes de una grabación ya sea de audio y video para la posterior transmisión al auditorio, de otra manera se tendría que escuchar y visualizar una grabación completa cada vez que se quisiera repasar lo más trascendente de un determinado evento. Si pensamos en la transmisión de un programa de noticias donde se esté llevando a cabo una entrevista, el presentador podría confrontar el discurso actual del entrevistado con los discursos o entrevistas anteriores, localizando de manera rápida las citas de dicha persona.

De igual manera en el sector público, por ejemplo en instituciones como el Poder

Judicial donde a partir de la reforma constitucional aprobada en 2008 [Monroy et al., 2008] se implantó de un nuevo sistema penal llamado Juicio Oral [Montejano y Cordero, 2008, Rivera, 2011, Terrazas, 2010], es posible utilizar un sistema como el que se plantea. Las grabaciones de los interrogatorios así como las pruebas que consistan en material digital, al igual que en el caso anterior, podrían ayudar no solo para confrontar declaraciones en un contrainterrogatorio, o permitir dirigirse a declaraciones especificas en un proceso de desahogo de pruebas, pero también para recordar declaraciones y lograr un proceso fluido.

Un método de búsqueda como el que se desarrolló en este trabajo permite realizar un alineamiento tanto en línea como fuera de línea. Otra aplicación que surge en materia de combate a la delincuencia y tomando en cuenta la idea del enunciado anterior, es la posibilidad de realizar un monitoreo en flujos de audio, tales como conversaciones telefónicas. El objetivo sería la localización de palabras o frases de uso común entre delincuentes. En el caso más grave como el de terroristas, las palabras o frases como "bomba", "explosivo", etc., tendrían un interés significativo por lo que estas implican. El proceso de monitoreo de la mano con técnicas de clasificación de documentos, e.g., el modelo bolsa de palabras (*bag of words*), permitirían clasificar dichas conversaciones, por tanto sería posible entonces contar con un perfil basado en la gravedad de la conversación.

Es importante decir que las tareas que se mencionaron y donde el sistema propuesto nos ayudaría, son realizadas por un operador con un reproductor de audio y video, si la grabación está dividida por temas se puede avanzar o retroceder según sea el caso hasta dicho tema y localizar el fragmento deseado, de otra manera, se debe revisar la grabación desde que inicia y hasta donde se localice dicho fragmento. Lo cual resulta en un proceso largo y tedioso si pensamos que se va a buscar en una grabación de una hora o de más duración. Adicionalmente el proceso estaría limitado a realizarse antes de que se lleve a cabo el juicio o la transmisión del programa. Por tanto, respecto al programa de noticias, tendría que seguirse un guión muy estricto y sin improvisaciones debido a lo tardado del proceso. En cuanto al juicio, se perdería la agilidad inherente a dicho proceso.

1.3. Objetivos de la Tesis

1.3.1. Objetivo general

Implementar un sistema que sea capaz de buscar y ubicar una elocución de una palabra o frase en archivos de audio.

1.3.2. Objetivos particulares

- Obtener una representación en un espacio de dimensionalidad baja respecto a una señal de audio completa, que caracterice de manera única la elocución o el archivo de audio.
- Implementar esquemas de emparejamiento aproximado de cadenas y adecuarlos al problema que nos ocupa.

1.4. Descripción de Capítulos

En el capítulo 2 se presentan las bases teóricas para caracterizar una consulta y los modelos que nos van a permitir realizar la búsqueda de la misma.

En el capítulo 3 se explica detalladamente el diseño del sistema, los valores de las variables que se utilizaron y se muestra con algoritmos la implementación de cada fase de dicho sistema.

En el capítulo 4 se expone detalladamente los experimentos realizados, i.e., los valores que tomaron las variables del sistema y los fundamentos de porque se realizó cada experimento. En la segunda parte del capítulo se presentan y se analizan los resultados obtenidos después de ejecutar cada experimento.

En el capítulo 5 se dan las conclusiones a las que se llegaron después de haber analizado el desempeño del sistema y una comparativa con respecto al estado del arte. En la segunda etapa del capítulo se plantean los esquemas que se podrían abordar para mejorar el desempeño y como se realizaría la adaptación al sistema actual.

Capítulo 2

Estado del Arte

La búsqueda de elocuciones de palabras es una área que tiene como objetivo la localización de elocuciones de palabras o frases en flujos de audio. Las dos áreas principales en las que se han desarrollado enfoques que dan solución al problema que nos ocupa son las siguientes: *Keyword Spotting* (KWS) donde el problema consiste en localizar en un archivo las palabras clave que se introducen como consulta y *Spoken Term Detection* (STD) que tiene como objetivo regresar una lista de documentos calificados respecto a la relevancia a una consulta de voz. También Existen enfoques en los que se puede resolver nuestro problema de manera indirecta tales como: recuperación por contenido en grabaciones de voz, recuperación de información, indexamiento y recuperación, búsqueda de voz y recuperación de documentos de voz (SDR del Inglés *Spoken Document Retrieval*). Tal y como es de esperarse el conjunto de áreas que se acaban de mencionar tienen como base elementos que forman parte de un sistema de reconocimiento de voz, como son, la caracterización de la señal, técnicas de alineamiento y modelos usados en el reconocimiento de patrones.

2.1. Antecedentes

A partir de los trabajos realizados por Rabiner y Schafer en [Rabiner y Schafer, 1978] se han desarrollado esquemas robustos que han dado como resultado el fortalecimiento de la teoría de cada una de las líneas de investigación del tratamiento digital de señales de voz, como son, la síntesis de voz, el reconocimiento de voz y la síntesis mediante la conversión

de texto a voz, principalmente.

Un tema importante dentro del reconocimiento de voz [Rabiner y Juang, 1993] y que surgió gracias a los avances en dicha rama fue la necesidad de identificar documentos relevantes a una consulta dada, donde los documentos en los que se realiza la búsqueda y la consulta pueden ser originados desde dos medios, texto y voz.

En general el esquema dominante en las aplicaciones que se han realizado para resolver el problema del párrafo anterior, es el uso de un LVCSRS o de un reconocedor de fonemas, de la mano de algoritmos de recuperación de información (IR del Inglés *Information Retrieval*) [Glavitsch y Schäuble, 1992], [Schäuble y Wechsler, 1995], [Brown et al., 1996], [Choi et al., 1998], [Wechsler et al., 2000], [Olsson y Oard, 2009]. El LVCSRS se utiliza para hacer una transcripción completa de los documentos y en ocasiones de la consulta, usando palabras o fonemas como unidades básicas. Donde el problema de la recuperación de los documentos es llevada a cabo mediante el procesamiento de texto. Garofolo *et al.* en [Garofolo et al., 2000] resumen casos de éxito de este grupo de aplicaciones. En la sección anterior se mencionó que nuestro problema podría ser resuelto indirectamente y en efecto esto es posible usando un esquema como el que se mencionó anteriormente ya que se puede no solo localizar en que documentos ocurrió la consulta sino que podemos establecer en que posición del texto se encontró.

En los esquemas anteriores el reconocimiento de voz lo realiza un Modelo Oculto de Markov (HMM del Inglés *Hidden Markov Model*), existen enfoques alternativos al uso de un HMM, como son el uso de lógica difusa (*Fuzzy Logic*) como exponen Liu *et al.* en [Liu et al., 2002, Liu et al., 2004] y máquinas de soporte vectorial (SVM del Inglés *Support Vector Machine*) por Zhu *et al.* en [Zhu et al., 2007]. Estos enfoques además de realizar IR en los archivos de la base de datos, realizan una clasificación de archivos de audio, detectando cuando hay voz, música, silencio, sonido ambiental, voz con música, voz con sonido ambiental y clasificación por genero del hablante.

El uso de índices de proximidad (*Proximity Indexes*) para agilizar la búsqueda de elocuciones es un aspecto deseable en un sistema de este tipo, i.e., un sistema que trate con archivos multimedia. Dimitry y Bovbel en [Dmitry y Bovbel, 2007] presentan una estructura de datos llamada árbol celular (*Cellular Tree*) y un esquema de IR llamado consulta pro-

gresiva (*Progressive query*) que puede ser aplicado con el índice o sin él. Este procedimiento divide la base de datos en subconjuntos y en cada subconjunto realiza comparaciones individuales entre la consulta y cada elemento del subconjunto. Por tanto básicamente una subconsulta es un proceso de particionado donde el resultado de cada comparación es regresado y fusionado con los resultados que ya han sido obtenidos hasta ese instante. El indexamiento y la búsqueda son en general muy similares al de un índice basado en árboles [Burkhard y Keller, 1973]. En un índice basado en árboles los objetos son almacenados en cubos *buckets* que pueden ser implementados como arreglos o listas ligadas. En un árbol celular los objetos parecidos se conjuntan en células. Cada célula es conformada por un árbol de expansión mínima, por lo tanto los objetos cuando son insertados en una célula se insertan en un nodo de dicho árbol. En la búsqueda las operaciones son las mismas salvo el recorrido en el árbol de expansión mínima y la conceptualización bajo el esquema de la búsqueda progresiva. Singh *et al.* en [Singh et al., 2014] presentan un esquema de recuperación basado en la similaridad de documentos parecido al que aborda este trabajo de tesis. Los coeficientes cepstrales de Mel son extraídos tanto de los archivos de la base de datos como de la consulta, en seguida se usa cuantización vectorial (VQ de Inglés *Vector Quantization*) y el resultado es una secuencia de enteros. La búsqueda dentro del proceso de VQ se hace con un *k-d tree*. Después la similaridad entre el archivo y la consulta es determinado con LCS. Por lo tanto la principal diferencia con nuestro trabajo reside en la manera en que se aplica LCS.

Al igual que en SDR en KWS la mayoría de los trabajos realizados usan un LVCSR para convertir un documento completo de voz a texto o realizando un decodificación de la secuencia de estados más probable de un HMM producida por una secuencia de vectores o una secuencia de observaciones, dependiendo de si el HMM es discreto o continuo [Moyal et al., 2013], [Silaghi, 2005]. Moyal *et al.* en [Moyal et al., 2013] presentan un resumen de los paradigmas basados en un LVCSR. Los tres esquemas descritos, difieren en el tamaño del vocabulario usado y la transcripción de las unidades fonéticas (palabras, fonemas, etc.).

Li y Liao en [Li y Liao, 2012] exponen el uso del DTW y de modelos de mezcla de Gaussianas (GMM, del Inglés *Gaussian Mixture Model*), el esquema utiliza un conjunto de 10 ejemplos de una palabra o frase, esta palabra o frase es de hecho la palabra clave que se

desea encontrar. En el caso donde se utiliza DTW, el conjunto de entrenamiento sirve para realizar un alineamiento entre la palabra o frase que se utiliza en la prueba de búsqueda y cada uno de los ejemplos del conjunto de entrenamiento, al final las distancias obtenidas son promediadas. En el caso en el que se usa un GMM para llevar a cabo la búsqueda, el conjunto de 10 ejemplos es usado para entrenar el modelo, en la fase de búsqueda se calcula la Log verosimilitud de que una porción de la palabra de prueba esté contenida en el GMM. Tanto los ejemplos o conjunto de entrenamiento como la palabra de prueba son procesados por un modulo de extracción de características tradicional, i.e., dichas señales son procesadas marco por marco, de cada marco se obtienen 12 coeficientes cepstrales de Mel, aumentados con 12 coeficientes delta y 12 coeficientes delta-delta. Al final si el valor promedio de las distancias obtenidas usando DTW y el promedio de las log verosimilitudes, son menor y mayor respectivamente a un valor umbral se dice que hubo una ocurrencia.

Chen *et al.* en [Chen et al., 2015] proponen un método *query-by-example* que utiliza un esquema de extracción de características basado en una red LSTM del Inglés *long short-term memory* que es una red neuronal recurrente (RNN, del Inglés *recurrent neural network*). El proceso de extracción de características es aplicado únicamente en las regiones donde se encuentra voz. Para entrenar los modelos de las redes neuronales se utilizaron grabaciones de voz, en las cuales, es importante mencionar, no se tenían señaladas las palabras clave que se seleccionaron para ser buscadas. Para modelar una palabra clave que se pretendía buscar se usaron 3 elocuciones de dicha palabra y varias elocuciones más para hacer la prueba de búsqueda. Una vez que se tienen entrenadas las redes neuronales, los 3 ejemplos dados para modelar una palabra clave son procesados por el extractor de características, el vector de características obtenido de dicho proceso es un conjunto fijo de k valores que corresponden con los últimos k valores de activación de la segunda capa oculta de la red LSTM, posteriormente los 3 vectores son promediados para obtener un solo vector. En tiempo de ejecución la elocución de una palabra clave que se quiere buscar es procesada por el mismo modulo de extracción de características el cual produce un vector de tamaño fijo, dicho vector es comparado usando distancia coseno con el vector promediado del procedimiento descrito anteriormente. Después, se detecta o no dicha palabra clave dependiendo del valor de similitud obtenido.

Mamou *et al.* en [Mamou et al., 2007] presentan un esquema de STD basado en un sistema automático de reconocimiento voz y uno de IR. Un resumen de los sistemas que usan este paradigma lo exponen Fiscus *et al.* en [Fiscus et al., 2007]. Parada *et al.* en [Parada et al., 2009] así como Can y Saraclar en [Can y Saraclar, 2011] abordan el problema con un transductor de estados finitos, donde el autómata de dos cintas (una de entrada y una de salida) reconoce una cadena, si esta se encuentra en la cinta de entrada. Existen esquemas que plantean la solución al STD con el uso de algoritmos de alineamiento de segmentos cortos como el DTW [Chan y Lee, 2010, Chan y Lee, 2011]. Para realizar un alineamiento donde el comienzo y el final de un segmento que se va a alinear es aproximado [Mantena et al., 2014].

Shen *et al.* en [Shen et al., 2009] realizan una comparación de tres esquemas de alineamiento para la búsqueda de términos de voz (i.e., elocuciones de palabras o frases). Estos esquemas del tipo *query-by-example* son una alternativa a los esquemas que usan un sistema automático de reconocimiento de voz y los que realizan una búsqueda fonética. Las tres versiones presentadas son evaluadas con respecto a su precisión en la tarea de recuperación de elocuciones que contengan términos usados como consulta. La base de datos está formada por grabaciones de conversaciones telefonicas extraídas del corpus *Fisher*[1]. Tanto las elocuciones de la base de datos como las consultas son transcritas a *lattices* de fonemas mediante un reconocedor de fonemas basado en una red neuronal con arquitectura TRAP [Hermansky y Sharma, 1998]. En la fase todavía de entrenamiento se crea un índice fonético basado en n-gramas de las elocuciones usadas como base de datos. Por último en la fase de prueba, los *lattices* de las consultas son convertidos también en n-gramas para realizar un alineamiento con los n-gramas creados previamente. Los tres sistemas de recuperación presentados son los siguientes: 1)*Direct Index Matching*, 2)*Edit Distance Alignment System* y 3)*HMM Alignment System*. En el primer sistema se realiza una comparación entre los n-gramas de la consulta y de las grabaciones indexadas, sean Q y I respectivamente, moviendo la consulta sobre cada elocución emulando una ventana móvil. La función que califica el parecido entre cada columna consiste en el calculo de la probabilidades a posteriori asociadas a cada columna tanto de Q como de I. Para cada ventana móvil se calcula la probabili-

[1]https://www.ldc.upenn.edu

dad conjunta. Para determinar si la consulta está contenida en una elocución indexada se calcula la razón de verosimilitud con respecto de un modelo de fondo. El segundo sistema considera igualmente los n-gramas Q y I. Dada la distribución de los símbolos p o fonemas de un alfabeto A en una columna i del índice I, $P(p|I_i)$, la distancia $d_{\phi}^{Vit}(Q, I)$ como una generalización del algoritmo Viterbi, donde ϕ es la distancia de edición entre cadenas de símbolos en A, se calcula obteniendo el logaritmo negativo de la probabilidad de la secuencia de operaciones de edición más parecida entre cualquier camino a través de Q y su vez cualquiera a través de I. El tercer sistema de alineamiento se basa en un HMM discreto. En este sistema la consulta se modela como un HMM, cada columna del n-grama se representa como un estado del HMM y las probabilidades de los unigramas como probabilidades de observación. Para determinar la relevancia de la elocución con respecto de la consulta, se calcula la probabilidad completa del algoritmo Forward del índice I dado el modelo de la consulta λ_Q, $P(I_1|\lambda_Q)$.

Hazen *et al.* en [Hazen et al., 2009] presentan un método de tipo *query-by-example* con el que se busca encontrar una consulta en segmentos de elocuciones. La matriz de características acústicas que se usa en este esquema consiste en las distribuciones a posteriori a lo largo del tiempo de un conjunto de símbolos o fonemas. La búsqueda es realizada usando una variante del DTW para alinear los posteriogramas. El DTW modificado que se presenta permite encontrar el camino optimo de alineamiento, es decir la ocurrencia de la consulta dentro de una elocución de referencia. La primera restricción que se establece es que el DTW modificado no permite la extensión simultanea de caminos en segmentos de la consulta y la elocución. La segunda restricción consiste en favorecer extensiones de camino con duración similar, ponderando el valor de similitud por un factor de pendiente de alineamiento a lo largo de extensiones individuales de un camino hipotético. El factor de pendiente de alineamiento es ponderado exponencialmente por un valor φ que controla el tipo de la pendiente del camino de alineamiento, donde valores grandes de φ hacen que el algoritmo dé preferencia a caminos con una pendiente de 45$\circ$. Finalmente se extiende el esquema presentado agregando el procesamiento de varios ejemplos de una consulta, este esquema consiste en obtener valores de similitud de cada ejemplo y combinar dichos valores obtenidos en cada proceso de alineamiento, transformando dichos valores de similitud aplicándoles

una función exp y después promediando dichos valores. Esta extensión al sistema mejora notablemente la tasa de error de la búsqueda.

2.2. Marco Teórico

Camarena-Ibarrola *et al.* en [Camarena-Ibarrola y Chávez, 2006, Camarena-Ibarrola et al., 2009] presentan trabajos de investigación en la identificación de interpretaciones musicales y el monitoreo de estaciones de radio, dichas investigaciones son la base teórica para realizar la búsqueda en nuestro marco de trabajo.

El marco de trabajo consta primordialmente de dos procesos. En primera instancia el procesamiento digital de la señal de voz y en segunda las técnicas de alineamiento usadas en el ASM. Las dos diferencias entre este esquema y la mayoría de los trabajos resumidos en la sección anterior son, el uso de un esquema de extracción de características que no depende de realizar una transcripción fonética, evitando así el uso de reconocedores de fonemas o sistemas de reconocimiento de voz. En la parte de la búsqueda se reemplaza el uso del DTW calculado de manera tradicional, por técnicas de alineamiento que permiten realizar la búsqueda en una sola pasada sin necesidad de realizar un alineamiento cada que se sobreponga la consulta a una ventana de una elocución de referencia, evitando tanto el costo computacional que implica este procedimiento como el costo de memoria que implica almacenar una matriz de alineamiento.

2.2.1. Procesamiento digital de la señal de voz

El primer módulo de cualquier sistema de reconocimiento de voz es el procesamiento de la señal de voz. En la Figura 2.1 se muestra cada uno de los componentes del módulo de procesamiento digital de la señal de voz PDS y el flujo que lleva dicho procesamiento. La señal de voz es tratada en diferentes dominios. En el dominio del tiempo se realiza la segmentación de una elocución para reducir el error del reconocedor por alteraciones que puede causar el ruido al inicio y al final de la grabación. En un segundo plano la señal es

procesada en los dominios de la frecuencia y la cuefrencia[2] de la mano de modelos basados en el sistema auditivo humano. El análisis descrito anteriormente permite extraer características cepstrales significativas de la señal que dan como resultado un patrón que puede ser identificado en un sistema de reconocimiento, al resultado de este análisis se le llama extracción de características.

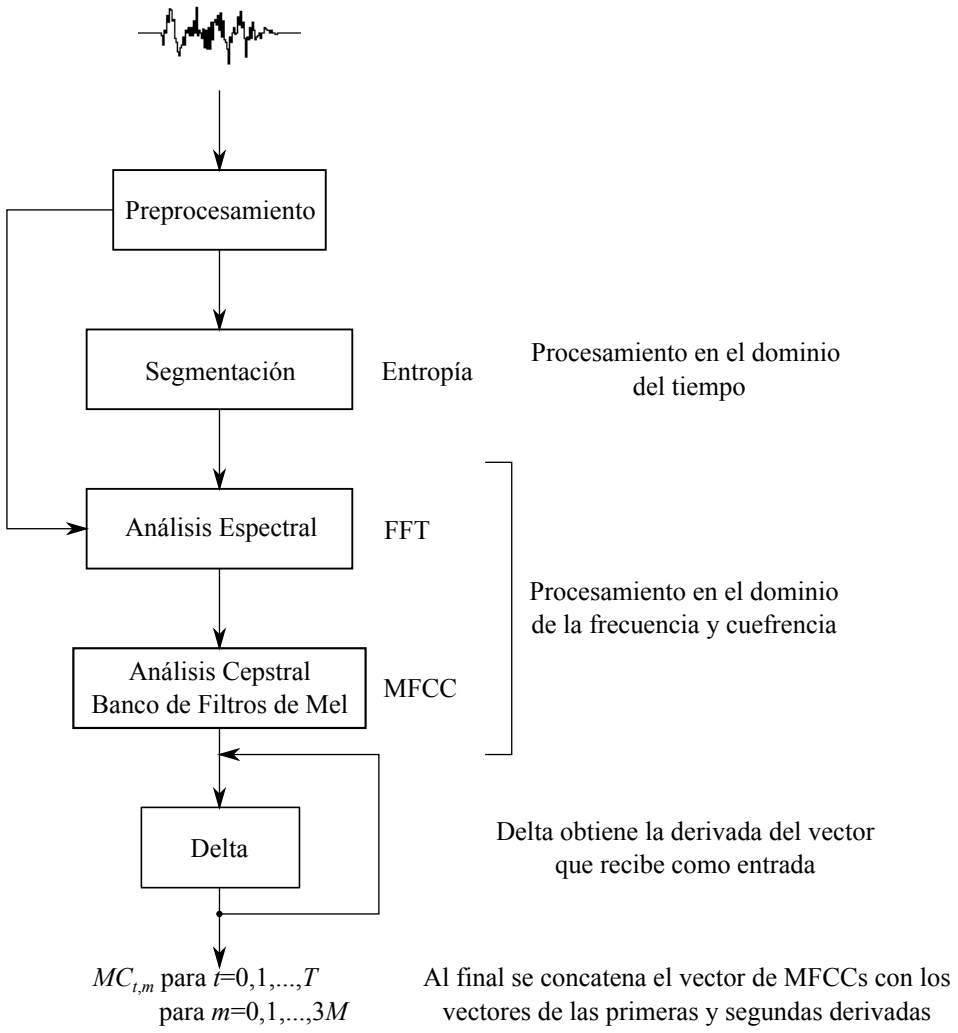


Figura 2.1: Flujo del procesamiento digital de la señal de voz.

[2]Los términos definidos en el procesamiento cepstral de la voz son tomados del espectral simplemente con algunas letras en otro orden, e.g., "espectral" corresponde con "cepstral", "espectro" con "cepstro", "frecuencia" con "cuefrencia", etc.

Preprocesamiento

En una etapa previa al tratamiento de la señal es útil aplicar un filtro de preénfasis que es un filtro FIR (del Inglés *Finite Impulse Response*) de primer orden, $y[n] = x[n] - \alpha x[n-1]$ donde $0.95 \leq \alpha \leq 0.99$ y tiene como objetivo amplificar las frecuencias altas, enfatizándolas para compensar el hecho de que el oído humano percibe mejor las bajas frecuencias [Huang et al., 2001], [Camarena, 2011].

Procesamiento de la señal en el dominio del tiempo

El análisis en el dominio del tiempo se realiza por ventanas de tiempo corto de la señal empezando desde el inicio de la grabación y avanzando 1/3 del tamaño de ventana w (típicamente de 30ms), tal y como se ilustra en la Figura 2.2. Por cada ventana se puede obtener la energía, la magnitud, el régimen de cruces por cero o la entropía. Al final por cada ventana y dependiendo del tipo de característica que se calculó se obtendrá la señal de energía, de entropía, etc.

Figura 2.2: Ventaneo de una señal de voz.

Al realizar una grabación es imposible sincronizar el momento en el que un hablante inicia una elocución y el momento en el que el sistema comienza a grabar. Para dejar de lado los instantes de la grabación que no contienen voz se tiene que segmentar la señal. La segmentación consiste en localizar cuando comienza y cuando termina una elocución.

En el dominio del tiempo el cómputo de la entropía de tiempo corto nos permite

segmentar la señal. Si el valor de la señal de entropía en el instante t es mayor que cierto valor umbral podemos decir que en ese instante comenzó la elocución. Para determinar el final de la elocución, aplicamos el proceso descrito anteriormente a partir del final de la grabación y avanzamos hacia el inicio de la misma.

Entropía de la señal de voz

La entropía mide el nivel de información en una señal. Camarena en [Camarena, 2011] da una explicación de la entropía de una señal de voz, a continuación haremos un recuento de los conceptos básicos necesarios para llevar a cabo el cálculo de esta.

Una señal captada por un solo canal de audio que es digitalizada a una frecuencia de muestreo de 8 kHz con un tamaño de muestra de 16 bits con signo, tiene un rango de valores de -32768 hasta 32767 cada uno de esos valores denotado por $v_1, v_2, \ldots, v_n$ tiene una probabilidad p_i de ocurrir, $p_i = P(v_i)$. El contenido de información de un valor v_i se define como $I(v_i) = \ln(\frac{1}{p_i})$. La información de una señal es el valor esperado del contenido de información de cada valor de la señal y está dada por $H = E[I] = \sum_{i=0}^{N-1} I(v_i)p_i = \sum_{i=0}^{N-1} p_i \ln(\frac{1}{p_i}) = -\sum_{i=0}^{N-1} p_i \ln(p_i)$, donde N es el número de valores diferentes que pueden tomar las muestras de la señal y $p_i = \frac{f_i}{N}$, donde f_i el número de ocurrencias del valor v_i.

Existen dos maneras de calcular la entropía de la señal, el método no paramétrico y el paramétrico. En el método no paramétrico para calcular la entropía es necesario contar con la función de densidad de probabilidad (PDF del Inglés *Probability Density Function*) que se obtiene determinando el histograma de las muestras de la ventana y después normalizando por el número de muestras de la misma. Por ultimo la entropía de ese segmento se obtiene con (2.1).

$$H = -\sum_{i=0}^{NB-1} p_i \ln(p_i) \tag{2.1}$$

donde NB es el número de *bins* del histograma.

En cuanto al método paramétrico, si asumimos que las muestras de cada marco se distribuyen normalmente y sabiendo que $\int p(x)dx = 1$ y $\sigma^2 = \int p(x)x^2dx$, la entropía de dicha distribución se obtiene como sigue.

$$
\begin{aligned}
p(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x^2)}{2\sigma^2}} \\
-\ln p(x) &= \ln\sqrt{2\pi\sigma^2} + \frac{x^2}{2\sigma^2} \\
H(x) &= -\int p(x)\ln p(x)dx \\
&= \int p(x)\ln\sqrt{2\pi\sigma^2}dx + \int p(x)\frac{x^2}{2\sigma^2}dx \\
&= \ln\sqrt{2\pi\sigma^2} + \frac{\sigma^2}{2\sigma^2} \\
&= \ln\sqrt{2\pi\sigma^2} + \ln\sqrt{e} \\
&= \ln\sqrt{2\pi\sigma^2 e}
\end{aligned}
$$

donde $\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x(i)-\mu)^2$ y $\mu = \frac{1}{N}\sum_{i=0}^{N-1} x(i)$.

Procesamiento homomórfico y análisis cepstral

Cepstrum

Oppenheim *et al.* en [Oppenheim et al., 1968], Rabiner y Schafer en [Rabiner y Schafer, 1978] y en trabajos posteriores presentados por Huang *et al.* en [Huang et al., 2001] y Schafer en [Schafer, 2008] se hace referencia al trabajo de Bogert, Healy y Tukey realizado en 1963, donde se acuñó el término *cepstrum*. El cepstrum de una señal digital se obtiene aplicando la Transformada Inversa Discreta de Fourier al logaritmo del espectro de magnitud. El espectro de magnitud es determinado previamente al aplicar la Transformada Discreta de Fourier a la señal, posteriormente obtener la magnitud del espectro y el espectro obtenido es escalado logarítmicamente. A continuación se muestran (2.2) y (2.3) que corresponden con las definiciones del cepstrum y el cepstrum complejo de una señal $x[n]$,

$$c[n] = \frac{1}{N}\sum_{n=0}^{N-1} \ln|X[k]| e^{\frac{j2\pi kn}{N}} \tag{2.2}$$

donde $X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$.

La variable independiente de $c[n]$ es la cuefrencia y su unidad es el segundo ya que es una señal discreta en el tiempo.

$$\hat{x}[n] = \frac{1}{N}\sum_{n=0}^{N-1}\left[\ln|X[k]| + j\mathbf{arg}[X[k]]\right] e^{\frac{j2\pi kn}{N}} \tag{2.3}$$

La operación definida en (2.3) fue llamada cepstrum complejo debido al uso del logaritmo complejo y para ser diferenciado del cepstrum.

Sistemas homomórficos

Oppenheim *et al.* en [Oppenheim et al., 1968] y posteriormente Rabiner *et al.* en [Rabiner y Schafer, 1978], Huang *et al.* en [Huang et al., 2001] y Schafer en [Schafer, 2008], presentan de manera resumida lo que Oppenheim *et al.* llamaron una generalización de la noción de filtrado lineal, que consiste en identificar los sistemas nolineales que pueden cumplir el principio de superposición general. Adicionalmente se muestran como ejemplo de esos sistemas problemas donde una señal $x[n]$ está compuesta por dos señales $x_1[n]$ y $x_2[n]$ que fueron combinadas por medio de una multiplicación o por una convolución.

Para definir dicho principio de superposición general Oppenheim *et al.* definen dos operaciones vectoriales genéricas que en este trabajo nombraremos $\circ$ y $\diamond$. El resultado de aplicar $\circ$ a dos señales, sean $x_1[n]$ y $x_2[n]$, da como resultado una combinación de las dos señales la cual expresaremos como $x[n] = x_1[n] \circ x_2[n]$. Sea $\diamond$ la regla que permite combinar señales con un escalar y L una transformación lineal. La generalización de la noción de filtrado lineal requiere que se cumplan las siguientes propiedades:

$$L\left[x_1[n] \circ x_2[n]\right] = L\left[x_1[n]\right] \circ L\left[x_2[n]\right] \tag{2.4}$$

$$L\left[a \diamond x[n]\right] = a \diamond L\left[x[n]\right] \tag{2.5}$$

Las entradas de estos sistemas son vectores de un espacio vectorial, por lo tanto si hacemos que la regla $\circ$ corresponda con la suma vectorial y $\diamond$ con el producto escalar podemos ver que se trata del principio de superposición de un sistema lineal. Cualquier sistema que cumpla con (2.4) y (2.5) es llamado un sistema homomórfico y su representación canónica se muestra en la Figura 2.3.

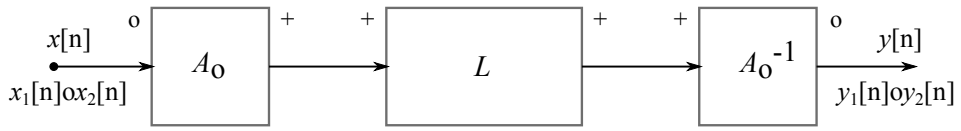


Figura 2.3: Representación canónica de un filtro homomórfico.

Se observa que a la entrada de $A_\circ$ se recibe $x[n] = x_1[n] \circ x_2[n]$ y a la salida se obtiene $A_\circ[x_1[n]] + A_\circ[x_2[n]]$ esta es una propiedad muy importante de los sistemas homomórficos de señales como la voz. $A_\circ$ también tiene asociada la propiedad de producto escalar, recibe $x[n]$ y a la salida se obtiene $a \diamond A_\circ[x[n]]$. El bloque intermedio L es un sistema lineal y $A_\circ^{-1}$ es la inversa de $A_\circ$, i.e., $A_\circ^{-1}[A_\circ[x[n]]] = x[n]$.

Ahora bien si tomamos en cuenta el sistema de producción de voz donde la señal producida es el resultado de la convolución de un tren de impulsos (en caso de sonidos vocalizados) o ruido blanco (en caso de sonidos no vocalizados) con el filtro de polos del modelo del tracto vocal. Oppenheim *et al.* en [Oppenheim et al., 1968] plantean la posibilidad de filtrar una señal de voz con el fin de analizar sus componentes correspondientes al tracto vocal. A los sistemas donde se sustituye la operación genérica $\circ$ por $*$ que corresponde con la operación de convolución, Oppenheim *et al.* los nombran sistemas de filtrado homomórfico para señales convolucionadas y en la Figura 2.4 se ilustra su representación canónica.

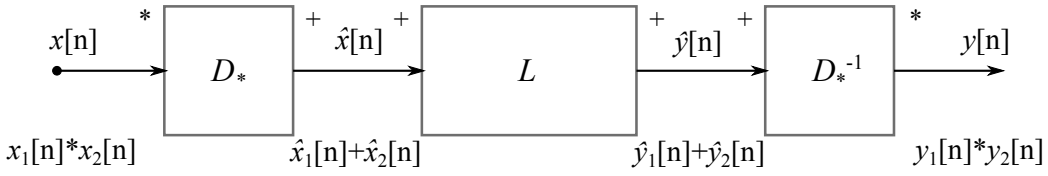


Figura 2.4: Representación canónica de un filtro homomórfico para señales convolucionadas.

Se puede observar que a la entrada del sistema característico D_* se recibe $x[n]$ que es la señal resultante de una convolución de dos señales $x_1[n] * x_2[n]$. Aplicando la transformación D_* a dicha señal se obtiene $D_*[x_1[n]] + D_*[x_2[n]]$, escrito de otra manera $\hat{x}_1[n] + \hat{x}_2[n]$. Al igual que en la representación canónica del filtro homomórfico general, L es un sistema lineal y D_*^{-1} es la inversa de D_* es decir las señales que están combinadas aditivamente al pasar por D_*^{-1} vuelven a estar combinadas por la convolución. La deducción

de este tipo de sistemas presentada en Oppenheim *et al.* tiene una fuerte relación con el cepstrum.

Existen otras aplicaciones en las que se desea hacer un filtrado de una señal que es el resultado de la multiplicación de dos señales [Oppenheim et al., 1968], como la modulación de amplitud, control automático de ganancia, entre otras aplicaciones. Si la operación $\circ$ es reemplazada por el producto punto $\cdot$ y se utiliza la operación potencia escalar como regla complementaria. Para que se cumplan las dos reglas de los sistemas homomórficos se necesita que al aplicarle una transformación T al producto de dos señales se obtenga la suma de las transformaciones individuales de las señales y que al aplicarle la misma transformación T al resultado de elevar a una potencia escalar una señal se obtenga el producto escalar de la transformación de dicha señal, en (2.6) y (2.7) se definen las sentencias anteriores.

$$T[x_1[n] \cdot x_2[n]] = T[x_1[n]] + T[x_2[n]] \tag{2.6}$$

$$T[(x[n])^a] = aT[x[n]] \tag{2.7}$$

donde T es la función ln.

A partir de las definiciones anteriores, el cálculo del cepstrum y el cepstrum complejo mediante un sistema homomórfico, el cual es equivalente a (2.2) y (2.3), corresponde con el que se plantea en la Figura 2.5.

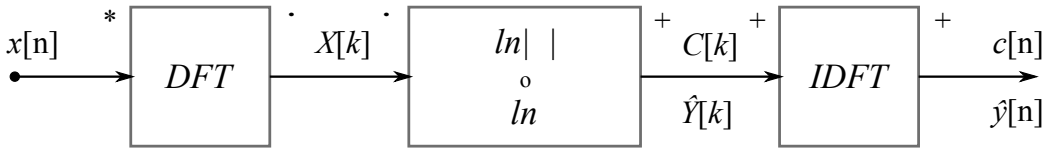


Figura 2.5: Representación canónica de un filtro homomórfico para señales convolucionadas.

La entrada de dicho sistema es la señal de voz $x[n]$, que es producida por la convolución de un tren de impulsos y los coeficientes del filtro de polos que modelan al tracto vocal. A $x[n]$ se le aplica la Transformada Discreta de Fourier para obtener el espectro de frecuencias $X[k]$. Posteriormente se le aplica el logaritmo a la magnitud del espectro o simplemente se le aplica el logaritmo a dicho espectro, ya sea para obtener el cepstro o el

cepstro complejo respectivamente. Finalmente se aplica la Transformada Inversa de Fourier para obtener el cepstrum o el cepstrum complejo.

Cepstrum de señales de voz

El análisis cepstral y espectral se lleva a cabo por ventanas al igual que cuando se procesa una señal en el dominio del tiempo. El tamaño de ventana corresponde con el número potencia de dos más cercano y mayor al número de muestras que hay en 30ms de una señal muestreada a 8 kHz y el paso entre ventanas es de 10ms. A diferencia del procesamiento por ventanas aplicado cuando se procesa la señal en el dominio del tiempo, el proceso que estamos describiendo y que se ilustra en la Figura 2.8, se lleva a cabo dentro de las marcas de segmentación como se puede apreciar en la figura. La primera ventana comienza a partir de la muestra que corresponde con el comienzo de la señal de voz, se avanzan 10ms y se repite el procedimiento hasta que se pueda procesar la última ventana completa sin exceder la marca de segmentación. El fin de procesar la señal en segmentos cortos de tiempo también llamados marcos es debido a que en ese lapso la señal se considera estacionaria, i.e., que los componentes de frecuencia no presentan cambios significativos [Camarena, 2011]. Posteriormente como se ilustra en la Figura 2.8 cada marco se multiplica por una ventana de Hamming, definida como $h[n] = 0.54 - 0.46\cos[2\pi n/N]$, para evitar el efecto de escurrimiento (*leakage*) debido a que los valores al inicio y al final son diferentes de cero y diferentes entre sí.

Tal y como se enuncia en los trabajos de Rabiner y Schafer en la práctica el cepstrum y el cepstrum complejo se calculan por medio de (2.2) y (2.3) respectivamente, usando la Transformada Discreta de Fourier y la Transformada Inversa Discreta de Fourier. Por tanto (2.2) se redefine en (2.8) y (2.3) se redefine en la (2.9). Dado que $X_s[k] = \sum_{n=0}^{N-1} x[n]\mathrm{e}^{\frac{-j2\pi nk}{N}}$, $0 \leq k < N$.

$$c_s[n] = \frac{1}{N}\sum_{k=0}^{N-1} C_s[k]\mathrm{e}^{\frac{j2\pi nk}{N}},\ 0 \leq n < N \tag{2.8}$$

donde $C_s[k] = \ln|X_s[k]|$, $0 \leq k < N$.

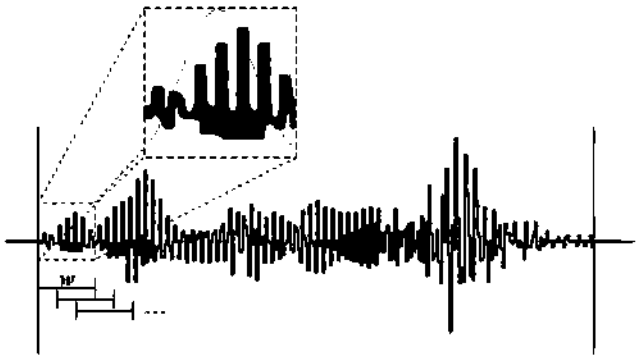


Figura 2.6: Ventaneo de una señal de voz.

$$\hat{x}_s[n] = \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}_s[k] \mathrm{e}^{\frac{j2\pi nk}{N}},\ 0 \leq n < N \tag{2.9}$$

donde $\hat{X}_s[k] = \ln|X_s[k]| + j\mathbf{arg}[X_s[k]], 0 \leq k < N$.

Las versiones del cepstrum redefinidas son debidas al procesamiento por ventanas de tiempo corto. Por tanto N corresponde al tamaño de ventana y el subíndice s denota precisamente un cepstrum nuevo.

Escala de Mel y MFCC

Un modelo que ha tenido mucho éxito en el procesamiento de señales de voz es la extracción de los coeficientes cepstrales de Mel o MFCC. Los MFCC están basados en el cepstrum pero difieren de este, ya que están escalados en frecuencia por un banco de filtros triangulares equiespaciados en la escala de Mel, en la Figura 2.7 se muestra un ejemplo de 13 filtros donde la frecuencia en mels mínima y máxima están fijadas a 121.9560 y 2146.064 respectivamente. Es importante decir que existen varias maneras de extraer los MFCC [Sigurdsson et al., 2006] y como resultado de cada implementación se obtienen bancos de filtros que tienen diferente forma entre si. En el caso del banco de filtros que se muestra en la Figura 2.7 podemos ver que los filtros colocados en las frecuencias bajas tienen una banda más estrecha que los filtros colocados en frecuencias altas y todos los filtros tienen una amplitud de 1. La escala de Mel presentada por Stevens *et al.* en [Stevens et al., 1937],

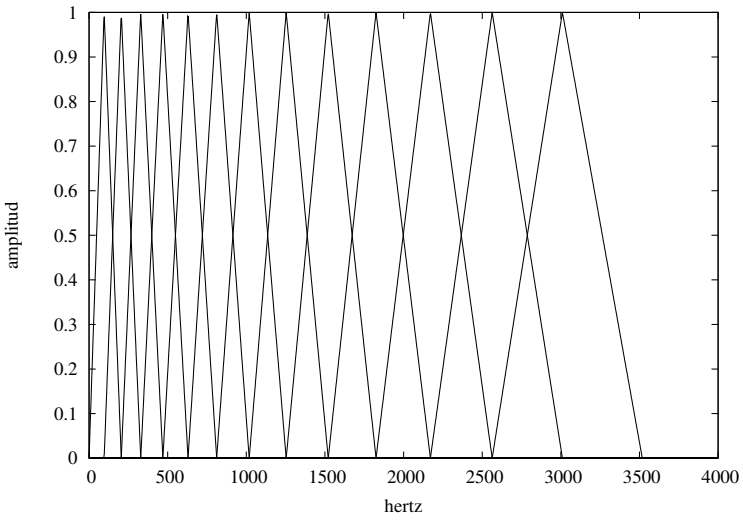


Figura 2.7: Banco de filtros triangulares de Mel.

es una escala perceptual basada en el sistema auditivo humano, debido a que se determinó experimentalmente por un número de oyentes. En la Figura 2.8 se muestra la equivalencia entre la unidad subjetiva mels y hertz.

Coeficientes Delta y Delta-Delta

Una característica muy importante de la voz humana es la variabilidad temporal del espectro de sus sonidos. La coarticulación modifica el patrón espectral de un sonido haciendo que este no consiga alcanzar el objetivo acústico, a este efecto se le llama subimpulso. La producción y percepción de un sonido deformado por dicho efecto puede llegar a ser afectado de tal manera que un sonido pequeño puede sonar como otro diferente [Furui, 1986]. Furui en [Furui, 1986] menciona que para compensar dichos efectos se aplica un sobre impulso basado en las características dinámicas espectrales de la señal de voz, que es el proceso que aplica el sistema auditivo humano para la percepción de dichas señales. El modelo en el que se basa Furui para enfatizar las características dinámicas de la señal de voz consiste en añadir a los valores de transición de los formantes la primera y segunda derivada.

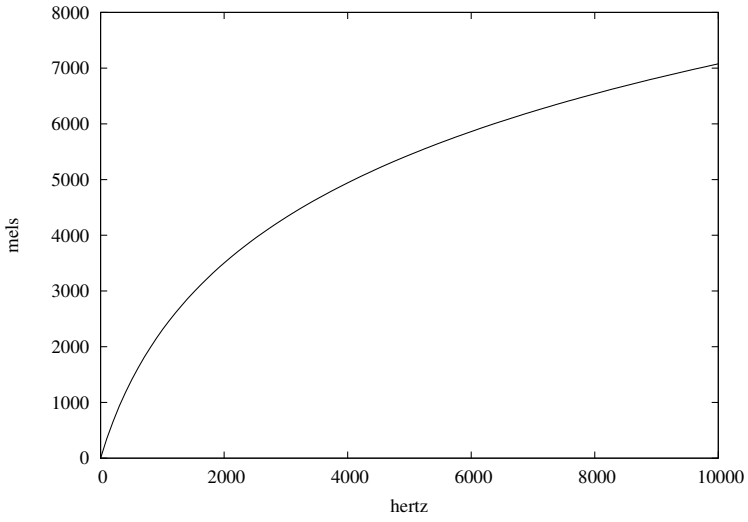


Figura 2.8: Escala de Mel.

2.2.2. Técnicas de alineamiento

Emparejamiento aproximado de cadenas

Como se mencionó anteriormente el problema del ASM consiste en encontrar un patrón en un texto cuando se permiten un número determinado de diferencias entre el patrón y su ocurrencia en el texto [Navarro y Raffinot, 2002]. Los modelos que se abordan en esta sección son la distancia de Levenshtein o distancia de edición (del Inglés *Edit Distance*) y LCS que es un caso particular de la distancia de Levenshtein.

Distancia de Edición o Distancia de Levenshtein

Para obtener la distancia entre un par de cadenas de texto primero definimos tres operaciones de edición: inserción, borrado y substitución. La distancia se define entonces como el número de operaciones de edición que se aplican a una cadena para convertirla en la otra y viceversa.

El cálculo de la distancia se lleva a cabo con programación dinámica. Cada celda de la matriz de programación dinámica $A_{i,j}$ contienen el número mínimo de operaciones de edición para emparejar $r_{1...i}$ con $s_{1...j}$. Al final la distancia entre r y s es el valor de $A_{|r|,|s|}$,

en (2.10), (2.11) y (2.12) se define cómo se obtiene la distancia de edición entre r y s.

$$A_{i,0} = i \quad \text{para} \quad i = 0, 1, \ldots, |r| \tag{2.10}$$

$$A_{0,j} = j \quad \text{para} \quad j = 0, 1, \ldots, |s| \tag{2.11}$$

$$A_{i,j} = \begin{cases} A_{i-1,j-1} & \text{si } r_i = s_j \\ \text{mín}\,(A_{i-1,j-1}, A_{i-1,j}, A_{i,j-1}) + 1 & \text{de otro modo} \end{cases} \tag{2.12}$$

para $i = 1, 2, \ldots, |r|$ y $j = 1, 2, \ldots, |s|$.

En la Tabla 2.1 se muestra un ejemplo del computo de la distancia de edición entre dos cadenas $r =$"prosa" y $s =$"programa".

Tabla 2.1: Ejemplo del cálculo de la distancia de edición entre las cadenas $r =$"prosa" y $s =$"programa".

			$\xrightarrow{j}$							
	r \ s	ε	p	r	o	g	r	a	m	a
	ε	0	1	2	3	4	5	6	7	8
	p	1	0	1	2	3	4	5	6	7
$i\downarrow$	r	2	1	0	1	2	3	4	5	6
	o	3	2	1	0	1	2	3	4	5
	s	4	3	2	1	1	2	3	4	5
	a	5	4	3	2	2	2	2	3	**4**

LCS

LCS es un caso particular de la distancia de edición en el que las únicas operaciones de edición son la inserción y el borrado. Entonces el computo de la LCS entre dos cadenas r y s es como sigue.

$$A_{i,0} = i \quad \text{para} \quad i = 0, 1, \ldots, |r| \tag{2.13}$$

$$A_{0,j} = j \quad \text{para} \quad j = 0, 1, \ldots, |s| \tag{2.14}$$

$$A_{i,j} = \begin{cases} A_{i-1,j-1} & \text{si } r_i = s_j \\ \text{mín}\,(A_{i-1,j}, A_{i,j-1}) + 1 & \text{de otro modo} \end{cases} \tag{2.15}$$

para $i = 1, 2, \ldots, |r|$ y $j = 1, 2, \ldots, |s|$.

En la Tabla 2.2 se muestra el calculo de la LCS entre las cadenas $r =$"prosa" y $s =$"programa", del ejemplo anterior.

Tabla 2.2: Ejemplo del cálculo de LCS entre las cadenas $r =$"prosa" y $s =$"programa".

			$\xrightarrow{j}$							
	r \ s	ε	p	r	o	g	r	a	m	a
	ε	0	1	2	3	4	5	6	7	8
	p	1	0	1	2	3	4	5	6	7
$i\downarrow$	r	2	1	0	1	2	3	4	5	6
	o	3	2	1	0	1	2	3	4	5
	s	4	3	2	1	2	3	4	5	6
	a	5	4	3	2	3	4	3	4	**5**

twLCS

Algunas técnicas del ASM han sido utilizadas en la recuperación de música. Guo y Siegelmann en [Guo y Siegelmann, 2004] abordan un esquema donde la consulta consiste en la interpretación (canto) de un segmento de una canción por una persona y la base de datos en la que se va a buscar está conformada por música monofónica. Por tanto la consulta es procesada de tal manera que se obtengan un conjunto de valores sucesivos en el tiempo llamada cadena de símbolos para poder realizar un alineamiento entre dicha cadena y la cadena de símbolos que componen las muestras de una canción en la base de datos.

El algoritmo twLCS presentado por Guo y Siegelmann en [Guo y Siegelmann, 2004] extiende LCS tomando ventaja de las características del DTW (*Dynamic Time Warping*)

[Sakoe y Chiba, 1978], el cual permite alinear secuencias que presentan variabilidad en el tiempo. El cómputo de twLCS entre dos cadenas r y s se define en (2.16), (2.17) y (2.18).

$$A_{i,0} = i \quad \text{para} \quad i = 0, 1, \ldots, |r| \tag{2.16}$$

$$A_{0,j} = j \quad \text{para} \quad j = 0, 1, \ldots, |s| \tag{2.17}$$

$$A_{i,j} = \begin{cases} \text{mín}\,(A_{i,j-1}, A_{i-1,j-1}, A_{i-1,j}) & \text{si } r_i = s_j \\ \text{mín}\,(A_{i,j-1}, A_{i-1,j}) + 1 & \text{de otro modo} \end{cases} \tag{2.18}$$

para $i = 1, 2, \ldots, |r|$ y $j = 1, 2, \ldots, |s|$.

Para ejemplificar el cálculo del twLCS, de la misma manera que se hizo con las técnicas anteriores, en la Tabla 2.3 se muestra la obtención de la distancia entre las cadenas $r =$"prosa" y $s =$"programa".

Tabla 2.3: Ejemplo del cálculo de twLCS entre las cadenas $r =$"prosa" y $s =$"programa".

			$j\rightarrow$							
	r \ s	ε	p	r	o	g	r	a	m	a
	ε	0	1	2	3	4	5	6	7	8
	p	1	0	1	2	3	4	5	6	7
$i\downarrow$	r	2	1	0	1	2	2	3	4	5
	o	3	2	1	0	1	2	3	4	5
	s	4	3	2	1	2	3	4	5	6
	a	5	4	3	2	3	4	3	4	**4**

Comparación entre la distancia de edición, LCS y twLCS

Con el objetivo de comparar los resultados que arrojan la distancia de edición, LCS y twLCS, a continuación se muestran las matrices de programación dinámica que se obtienen de cada algoritmo cuando se toma la distancia entre las cadenas $r_1 =$"4567" y $s_1 =$"41516171", $r_2 =$"4567" y $s_2 =$"44556677", $r_3 =$"4567" y $s_3 =$"4455661111177"; ver Tablas 2.4, 2.5 y 2.6 respectivamente.

Tabla 2.4: Cálculo de la distancia entre las cadenas r_1 ="4567" y s_1 ="41516171".

(a) *Distancia de edición*

					$\xrightarrow{j}$					
r_1 \ s_1		ε	4	1	5	1	6	1	7	1
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	1	1	2	3	4	5	6
	6	3	2	2	2	2	2	3	4	5
	7	4	3	3	3	3	3	3	3	**4**

(b) *LCS*

					$\xrightarrow{j}$					
r_1 \ s_1		ε	4	1	5	1	6	1	7	1
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	2	1	2	3	4	5	6
	6	3	2	3	2	3	2	3	4	5
	7	4	3	4	3	4	3	4	3	**4**

(c) *twLCS*

					$\xrightarrow{j}$					
r_1 \ s_1		ε	4	1	5	1	6	1	7	1
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	2	1	2	3	4	5	6
	6	3	2	3	2	3	2	3	4	5
	7	4	3	4	3	4	3	4	3	**4**

En el ejemplo de alineamiento realizado en las Tablas 2.4(a), 2.4(b) y 2.4(c), las tres técnicas usadas identifican correctamente los valores enteros que no ocurren en ambas secuencias. El valor obtenido al terminar el alineamiento corresponde con el número de operaciones que fueron utilizadas para convertir la cadena r_1 en la cadena s_1 y viceversa, i. e., 4 operaciones de borrado para que s_1 sea igual r_1 y en el otro caso 4 operaciones de inserción para que r_1 sea igual a s_1.

En el caso del ejemplo que se muestra en las Tablas 2.5(a), 2.5(b) y 2.5(c), podemos observar que para la distancia de edición y para LCS la distancia entre las cadenas r_2 y s_2 es de 4 debido a las cuatro operaciones de edición que se necesitan para que, en ambos casos, una cadena esté conformada por los mismos valores enteros que la otra. Sin embargo para el caso de twLCS que realiza un registro entre las dos cadenas el resultado es que las dos cadenas son iguales, la única diferencia entre las dos es que s_2 es una versión estirada en el tiempo de r_2 y al contrario r_2 es una versión compacta de s_2 en la que no se repiten los elementos a lo largo del tiempo.

Finalmente el ejemplo presentado en las Tablas 2.6(a), 2.6(b) y 2.6(c) donde se alinean las cadenas r_3 y s_3, observamos que tanto la distancia de edición como LCS acumulan un total de 9 operaciones de edición para convertir r_3 en s_3 y viceversa. Este resultado nos dice que estas cadenas son muy diferentes entre sí. En cambio twLCS únicamente inserta o borra los números uno que son los elementos de s_3 que no están en r_3.

Búsqueda de Texto

La búsqueda de un patrón q en cualquier posición de un texto o se logra procesando cada caracter del texto de referencia (texto donde se buscará) y asumiendo que es el comienzo de una nueva cadena. Tanto en la cadena que es usada como patrón de búsqueda como en el texto que corresponde con el patrón de referencia, existe un caracter que precede a todos los caracteres que conforman dichos patrones, este es el caracter vacío denotado por ε. La distancia de este con cada uno de los caracteres del patrón buscado comienza en 0 y aumenta en 1 cada vez que se compara con el siguiente caracter. Por otra parte no es necesario mantener en memoria la matriz de programación completa. El cómputo se puede llevar a cabo manteniendo únicamente una columna. El último valor de la columna que

Tabla 2.5: Cálculo de la distancia entre las cadenas r_2 ="4567" y s_2 ="44556677".

(a) *Distancia de edición*

						$\xrightarrow{j}$				
	r_2 \ s_2	ε	4	4	5	5	6	6	7	7
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	1	1	2	3	4	5	6
	6	3	2	2	2	2	2	3	4	5
	7	4	3	3	3	3	3	3	3	**4**

(b) *LCS*

						$\xrightarrow{j}$				
	r_2 \ s_2	ε	4	4	5	5	6	6	7	7
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	2	1	2	3	4	5	6
	6	3	2	3	2	3	2	3	4	5
	7	4	3	4	3	4	3	4	3	**4**

(c) *twLCS*

						$\xrightarrow{j}$				
	r_2 \ s_2	ε	4	4	5	5	6	6	7	7
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	0	1	2	3	4	5	6
$i\downarrow$	5	2	1	1	0	0	1	2	3	4
	6	3	2	2	1	1	0	0	1	2
	7	4	3	3	2	2	1	1	0	**0**

Tabla 2.6: Cálculo de la distancia entre las cadenas r_3 ="4567" y s_3 ="4455661111177".

(a) *Distancia de edición*

									$\xrightarrow{j}$						
r_3	s_3	ε	4	4	5	5	6	6	1	1	1	1	1	7	7
	ε	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	4	1	0	1	2	3	4	5	6	7	8	9	10	11	12
$i\downarrow$	5	2	1	1	1	2	3	4	5	6	7	8	9	10	11
	6	3	2	2	2	2	2	3	4	5	6	7	8	9	10
	7	4	3	3	3	3	3	3	4	5	6	7	8	8	**9**

(b) *LCS*

									$\xrightarrow{j}$						
r_3	s_3	ε	4	4	5	5	6	6	1	1	1	1	1	7	7
	ε	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	4	1	0	1	2	3	4	5	6	7	8	9	10	11	12
$i\downarrow$	5	2	1	2	1	2	3	4	5	6	7	8	9	10	11
	6	3	2	3	2	3	2	3	4	5	6	7	8	9	10
	7	4	3	4	3	4	3	4	5	6	7	8	9	8	**9**

(c) *twLCS*

									$\xrightarrow{j}$						
r_3	s_3	ε	4	4	5	5	6	6	1	1	1	1	1	7	7
	ε	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	4	1	0	0	1	2	3	4	5	6	7	8	9	10	11
$i\downarrow$	5	2	1	1	0	0	1	2	3	4	5	6	7	8	9
	6	3	2	2	1	1	0	0	1	2	3	4	5	6	7
	7	4	3	3	2	2	1	1	2	3	4	5	6	5	**5**

está siendo calculada nos da la distancia. En (2.19) y (2.20) se expresa este procedimiento usando la distancia de edición, donde q es la cadena utilizada como patrón de búsqueda, o es un texto en el que se buscará, C_i es la columna de alineamiento que se llenó un instante anterior y C'_i es la columna de alineamiento que se está llenando actualmente.

$$C_i = i \quad \text{para} \quad i = 0, 1, \ldots, |q| \tag{2.19}$$

$$C'_i = \begin{cases} C_{i-1} & \text{si } q_i = o_j \\ \text{mín}\left(C_{i-1}, C'_{i-1}, C_i\right) + 1 & \text{de otro modo} \end{cases} \tag{2.20}$$

para $i = 1, 2, \ldots, |q|$ y $j = 1, 2, \ldots, |o|$.

La versión de LCS para búsqueda en texto es similar a la de la distancia de edición, únicamente se deja de tomar en cuenta la operación de substitución en mín $\left(C_{i-1}, C'_{i-1}, C_i\right)+$ 1. El cálculo de LCS para realizar búsquedas en texto se define en (2.21) y (2.22).

$$C_i = i \quad \text{para} \quad i = 0, 1, \ldots, |q| \tag{2.21}$$

$$C'_i = \begin{cases} C_{i-1} & \text{si } q_i = o_j \\ \text{mín}\left(C'_{i-1}, C_i\right) + 1 & \text{de otro modo} \end{cases} \tag{2.22}$$

para $i = 1, 2, \ldots, |q|$ y $j = 1, 2, \ldots, |o|$.

Al igual que la distancia de edición y LCS, twLCS puede ser usada para realizar búsqueda en textos. En (2.23) y (2.24) se expresa el algoritmo modificado que permite realizar dicha tarea.

$$C_i = i \quad \text{para} \quad i = 0, 1, \ldots, |q| \tag{2.23}$$

$$C'_i = \begin{cases} \text{mín}\left(C_i, C_{i-1}, C'_{i-1}\right) & \text{si } q_i = o_j \\ \text{mín}\left(C_i, C'_{i-1}\right) + 1 & \text{de otro modo} \end{cases} \tag{2.24}$$

para $i = 1, 2, \ldots, |q|$ y $j = 1, 2, \ldots, |o|$.

Como ejemplo, en la Tabla 2.7 se muestra el computo de la búsqueda del patrón $q =$"con" en el texto $o =$"con el conocer", para cada una de las técnicas presentadas hasta el

momento. Como podemos ver los valores que están resaltados indican las posiciones donde se encontró el patrón. Si se revisan las ultimas posiciones de cada columna se puede verificar que estos valores obtenidos son los menores. El 0 obtenido en cada una de las posiciones donde hubo un emparejamiento significa que el patrón buscado se encontró en dos ocasiones sin tener que llevar a cabo ninguna operación de edición, i.e., ocurrió un emparejamiento perfecto.

Tabla 2.7: Ejemplo de la búsqueda del patrón $q =$"con" en el texto $o =$"con el conocer".

(a) *Distancia de edición*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
o	2	1	0	1	2	2	2	2	1	0	1	1	1	1	2
n	3	2	1	**0**	1	2	3	3	2	1	**0**	1	2	2	2

(b) *LCS*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
o	2	1	0	1	2	2	2	2	1	0	1	1	1	2	2
n	3	2	1	**0**	1	2	3	3	2	1	**0**	1	2	3	3

(c) *twLCS*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
o	2	1	0	1	2	2	2	2	1	0	1	1	1	2	2
n	3	2	1	**0**	1	2	3	3	2	1	**0**	1	2	3	3

En la Tabla 2.8 se presenta un ejemplo de búsqueda en texto usando la distancia de edición, LCS y twLCS, donde el patrón que se desea localizar es la cadena $q =$"cccooooonn" en el texto $o =$"con el conocer". La distancia de edición y LCS (ver Tablas 2.8(a) y 2.8(b)) localizan correctamente el patrón buscado, pero al mismo tiempo contabilizan 6 operaciones de edición necesarias para emparejar "cccooooonn" y "con". El registro en el tiempo que realiza twLCS (ver Tabla 2.8(c)) nos hace pensar que dicha técnica puede ser muy útil particularmente para el tipo de problema que se aborda en este trabajo, en el cual se realiza un alineamiento entre dos señales de voz, estas señales pueden contener varias elocuciones de la misma palabra o frase y dichas elocuciones pueden ser versiones compactas o estiradas en el tiempo. En el ejemplo desarrollado en la Tabla 2.8(c) podemos observar precisamente que aun cuando la cadena "cccooooonn" es una versión estirada de la cadena "con", twLCS es capas de identificar la posición exacta donde termina la ocurrencia de dicha cadena con un emparejamiento perfecto.

DTW

El DTW es un método usado para alinear en el tiempo dos secuencias, e.g., muestras de una señal de voz [Sakoe y Chiba, 1978]. El DTW permite realizar un doblado en el tiempo entre estas dos secuencias que no tienen la misma duración para poder realizar el alineamiento. El DTW se calcula usando programación dinámica y toma como referencia las condiciones de frontera, donde se asegura que el inicio y el final del alineamiento entre ambas señales coincidan en la misma celda de la matriz de programación dinámica, sea $D_{i,j}$ dicha matriz, para $i = 0, 1, \ldots, |R|$ y $j = 0, 1, \ldots, |S|$, donde R y S son dos secuencias o series de tiempo. Además de las condiciones anteriores, el cálculo de $D_{i,j}$ se lleva a cabo mediante las restricciones locales preestablecidas. Decimos que hay tres posibles celdas precedentes a $D_{i,j}$ que son $D_{i-1,j}$, $D_{i,j-1}$, $D_{i-1,j-1}$. Donde, avanzar ya sea de $D_{i-1,j}$ o $D_{i,j-1}$ hacia $D_{i,j}$ tiene un costo de 1, y avanzar desde $D_{i-1,j-1}$ hacia $D_{i,j}$ tiene un costo de 2. En la Tabla 2.9 se ilustran tanto las condiciones de frontera como las restricciones locales. Las Ecuaciones (2.25)-(2.28) conforman la recurrencia para el cálculo de $D_{|R|-1,|S|-1}$, para dos secuencias R y S, donde $d_{i,j}$ es la distancia entre el i-ésimo elemento de R y el j-ésimo elemento de S.

Tabla 2.8: Ejemplo de la búsqueda del patrón q =“cccooonn” en el texto o =“con el conocer”.

(a) *Distancia de edición*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
c	2	1	1	2	2	2	2	2	1	1	2	2	1	1	2
c	3	2	2	2	3	3	3	3	2	2	2	3	2	2	2
o	4	3	2	3	3	4	4	4	3	2	3	2	3	3	3
o	5	4	3	3	4	4	5	5	4	3	3	3	3	4	4
o	6	5	4	4	4	5	5	6	5	4	4	3	4	4	5
o	7	6	5	5	5	5	6	6	6	5	5	4	4	5	5
n	8	7	6	5	6	6	6	7	7	6	5	5	5	5	6
n	9	8	7	**6**	6	7	7	7	8	7	**6**	6	6	6	6

(b) *LCS*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
c	2	1	2	2	2	2	2	2	1	2	2	2	1	2	2
c	3	2	3	3	3	3	3	3	2	3	3	3	2	3	3
o	4	3	2	3	4	4	4	4	3	2	3	3	3	4	4
o	5	4	3	4	5	5	5	5	4	3	4	3	4	5	5
o	6	5	4	5	6	6	6	6	5	4	5	4	5	6	6
o	7	6	5	6	7	7	7	7	6	5	6	5	6	7	7
n	8	7	6	5	6	7	8	8	7	6	5	6	7	8	8
n	9	8	7	**6**	7	8	9	9	8	7	**6**	7	8	9	9

Continúa en la siguiente página

Continuación

Tabla 2.8: Ejemplo de la búsqueda del patrón q =“cccooooonn” en el texto o =“con el conocer”. (*Continuación*)

(c) *twLCS*

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
c	2	0	1	2	2	2	2	2	0	1	2	2	0	1	2
c	3	0	1	2	3	3	3	3	0	1	2	3	0	1	2
o	4	1	0	1	2	3	4	4	1	0	1	1	1	2	3
o	5	2	0	1	2	3	4	5	2	0	1	1	2	3	4
o	6	3	0	1	2	3	4	5	3	0	1	1	2	3	4
o	7	4	0	1	2	3	4	5	4	0	1	1	2	3	4
n	8	5	1	0	1	2	3	4	5	1	0	1	2	3	4
n	9	6	2	**0**	1	2	3	4	5	2	**0**	1	2	3	4

$$D_{0,0} = d_{0,0} \tag{2.25}$$

$$D_{i,0} = d_{i,0} + D_{i-1,0} \quad para\ i = 0, 1, 2, \ldots, |R| - 1 \tag{2.26}$$

$$D_{0,j} = d_{0,j} + D_{0,j-1} \quad para\ j = 0, 1, 2, \ldots, |S| - 1 \tag{2.27}$$

$$D_{i,j} = \text{mín} \begin{cases} D_{i-1,j-1} + 2d_{i,j} \\ D_{i-1,j} + d_{i,j} \\ D_{i,j-1} + d_{i,j} \end{cases} \tag{2.28}$$

para $i = 1, 2, \ldots, |R| - 1$ y $j = 1, 2, \ldots, |S| - 1$.

En la Tabla 2.10 se muestra un ejemplo de cómo se realiza el alineamiento entre dos secuencias usando DTW. Las secuencias son R =4567 y S =44556677. Dado que la

Tabla 2.9: Condiciones de frontera y restricciones locales.

$D_{0,0}$					
		$D_{i-1,j-1}$	$D_{i-1,j}$		
		$D_{i,j-1}$	$D_{i,j}$		
					$D_{\|R\|-1,\|S\|-1}$

única diferencia entre las dos secuencias es que S es una versión estirada en el tiempo de R la distancia entre las dos secuencias es de 0.

Tabla 2.10: Cálculo de la distancia entre las secuencias R =4567 y S =44556677.

				$j\rightarrow$					
r_2	s_2	4	4	5	5	6	6	7	7
	4	0	0	1	1	2	2	3	3
$i\downarrow$	5	1	1	0	0	1	2	4	5
	6	2	3	1	1	0	0	1	2
	7	3	5	3	3	1	1	0	**0**

DTW modificado

Al igual que las técnicas de alineamiento anteriores, el DTW puede ser modificado para no tener que calcular $D_{i,j}$ completamente, sino llenando únicamente una columna y monitoreando el ultimo valor de dicha columna. La primera columna que en el caso de las técnicas anteriores se llenaba con una serie consecutiva de números enteros desde 0 hasta el tamaño del patrón consulta, en este caso se llena con las distancias entre el primer elemento del patrón de referencia y cada elemento del patrón de consulta, ver (2.29). Para evitar que la distancia aumente a lo largo del alineamiento, el primer elemento de la columna que se está llenando, se calcula con la distancia entre el j-ésimo elemento del patrón de referencia y el primer elemento del patrón de consulta, en lugar de colocar un 0 como se hace en las

técnicas anteriores. La recurrencia modificada se define en (2.29) y (2.30).

$$C_i = d_{i,0} \quad para\ i = 0, 1, 2, \ldots, |R| - 1 \tag{2.29}$$

$$C'_i = \text{mín} \begin{cases} C_{i-1} + 2d_{i,j} \\ C'_{i-1} + d_{i,j} \\ C_i + d_{i,j} \end{cases} \tag{2.30}$$

para $i = 1, 2, \ldots, |R| - 1$ y $j = 1, 2, \ldots, |S| - 1$.

La modificación del DTW permite realizar búsquedas de cadenas en texto. En la Tabla 2.11 se muestra como DTW indica que la cadena buscada acaba de ocurrir. Esto queda ilustrado en la parte inferior de la tabla, en cada instante donde se obtuvo un 0. Este instante corresponde con el caracter que sucede a la ocurrencia de la cadena que se está buscando en el texto. De nuevo la obtención de un 0 significa que hubo un emparejamiento perfecto, esto se debe al doblado en el tiempo entre secuencias que realiza el DTW.

Tabla 2.11: Ejemplo de la búsqueda del patrón q =“cccoooonn” en el texto o =“con el conocer”.

	c	o	n		e	l		c	o	n	o	c	e	r
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	0	0	12	11	67	2	9	67	0	12	11	12	0	2
c	0	0	12	22	78	4	11	76	0	12	22	23	0	2
o	12	0	12	23	89	6	13	78	0	12	23	34	0	2
o	12	12	0	1	80	16	9	88	12	0	1	1	12	10
o	12	24	0	1	80	26	12	88	24	0	1	1	13	20
o	12	24	0	1	80	36	15	91	36	0	1	1	13	23
n	11	23	0	1	80	46	18	94	48	0	1	1	13	23
n	11	22	1	**0**	78	55	20	96	59	1	**0**	1	12	21

Una variante para el cálculo de la distancia de edición

Para algunos problemas en los que se requiere llevar a cabo un alineamiento, resulta insuficiente usar el criterio de comparar elementos de dos secuencias y tomar decisiones dependiendo si dichos elementos son iguales o no. Por tanto si evitamos el paso en el que se comparan dos elementos, podemos usar la distancia entre dichos elementos como el costo de la operación de sustitución [Camarena-Ibarrola y Chávez, 2006].

En (2.31), (2.32) y (2.33) se muestra el cálculo de la variante de la distancia de edición entre dos secuencias R y S. A lo largo de este trabajo cuando hagamos referencia a este procedimiento alternativo para obtener la distancia de edición, usaremos el nombre distancia de Leveshtein o Levenshtein.

$$A_{i,0} = i \quad \text{para} \quad i = 0, 1, \ldots, |R| \tag{2.31}$$

$$A_{0,j} = j \quad \text{para} \quad j = 0, 1, \ldots, |S| \tag{2.32}$$

$$A_{i,j} = \text{mín} \begin{cases} A_{i-1,j-1} + d_{i,j} \\ A_{i-1,j} + 1 \\ A_{i,j-1} + 1 \end{cases} \tag{2.33}$$

para $i = 1, 2, \ldots, |R| - 1$ y $j = 1, 2, \ldots, |S| - 1$.

El cálculo de Levenshtein como medida de distancia entre dos secuencias R y S, se muestra en la Tabla 2.12. De la tabla podemos observar que tal como se esperaba, la distnacia que se obtiene es de 4 ya que Leveshtein contabiliza los elementos que se tienen que editar para que S se convierta en R y viceversa.

Tabla 2.12: Cálculo de la distancia entre las cadenas R =4567 y S =44556677.

			$j \rightarrow$							
r_2	s_2	ε	4	4	5	5	6	6	7	7
	ε	0	1	2	3	4	5	6	7	8
	4	1	0	1	2	3	4	5	6	7
$i\downarrow$	5	2	1	1	1	2	3	4	5	6
	6	3	2	2	2	2	2	3	4	5
	7	4	3	3	3	3	3	3	3	**4**

La versión de Levenshtein que permite realizar búsquedas de cadenas en textos se define en (2.34) y (2.35). En la Tabla 2.13 se muestra un ejemplo del cálculo de Levenshtein para buscar una cadena en un texto. Levenshtein localiza los instantes donde ocurre la cadena buscada, sin embargo a diferencia de DTW, Levenshtein entrega una distancia aproximada de 6 que corresponde con el número de operaciones de edición necesarias para convertir "cccooooonn" en "con" y viceversa.

$$C_i = i \quad \text{para} \quad i = 0, 1, \ldots, |q| \tag{2.34}$$

$$C'_i = \text{mín} \begin{cases} C_{i-1} + d_{i,j} \\ C'_{i-1} + 1 \\ C_i + 1 \end{cases} \tag{2.35}$$

para $i = 1, 2, \ldots, |q|$ y $j = 1, 2, \ldots, |o|$.

Tabla 2.13: Ejemplo de la búsqueda del patrón q ="cccooooonn" en el texto o ="con el conocer".

	ε	c	o	n		e	l		c	o	n	o	c	e	r
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	1	0	1	1	1	1	1	1	0	1	1	1	0	1	1
c	2	1	2	2	2	2	2	2	1	2	2	2	1	2	2
c	3	2	3	3	3	3	3	3	2	3	3	3	2	3	3
o	4	3	2	3	4	4	4	4	3	2	3	3	3	4	4
o	5	4	3	3	4	5	5	5	4	3	3	3	4	5	5
o	6	5	4	4	5	6	6	6	5	4	4	3	4	5	6
o	7	6	5	5	6	7	7	7	6	5	5	4	5	6	7
n	8	7	6	5	6	7	8	8	7	6	5	5	6	7	8
n	9	8	7	**6**	7	8	9	9	8	7	**6**	6	7	8	9

2.3. Conclusiones del Capítulo

En este capítulo se presentaron los modelos matemáticos para procesar la señal de voz además de técnicas de alineamiento utilizadas en ASM, recuperación de música y reconocimiento de voz.

Los vectores de características como los que se calculan en este trabajo, compuestos por los MFCC concatenados con los coeficientes delta y delta-delta, son el estado del arte en el reconocimiento de voz y permiten construir sistemas de reconocimiento de voz robustos.

Es posible utilizar las técnicas de alineamiento descritas en este capítulo haciendo una analogía entre el proceso de búsqueda de una cadena de texto en un texto y el proceso de búsqueda de la elocución de una palabra o frase en una grabación de audio.

Capítulo 3

Diseño e Implementación

En este capítulo se explica a detalle el funcionamiento del esquema de búsqueda de elocuciones de palabras o frases en archivos de audio. Este esquema queda descrito explicando la interacción que existe entre el módulo de procesamiento digital de la señal de voz y el de búsqueda en texto. Con la finalidad de ayudar en la clarificación del trabajo se presenta la implementación del sistema junto con diagramas de bloques con flujos de datos y el pseudocódigo de los métodos o procedimientos más relevantes.

3.1. Diseño del sistema

La caracterización tanto de los archivos de audio contenidos en la base de datos, como de la elocución usada para consultar y la identificación del instante en el que el patrón de búsqueda es más parecido a determinada secuencia en el patrón de referencia son los dos módulos principales que componen nuestro sistema. En la Figura 3.1 se ilustra el inicio del proceso de búsqueda que es precisamente donde queda definida la interacción entre los dos módulos.

Tal y como se muestra en la parte superior de la Figura 3.1, se cuenta con una base de datos de grabaciones, de las cuales se obtendrán sus matrices de características y estas a su vez conformarán otra base de datos, este procedimiento es el primer paso antes de realizar una consulta. El propósito de este paso previo es no agregar un costo computacional extra al proceso de búsqueda.

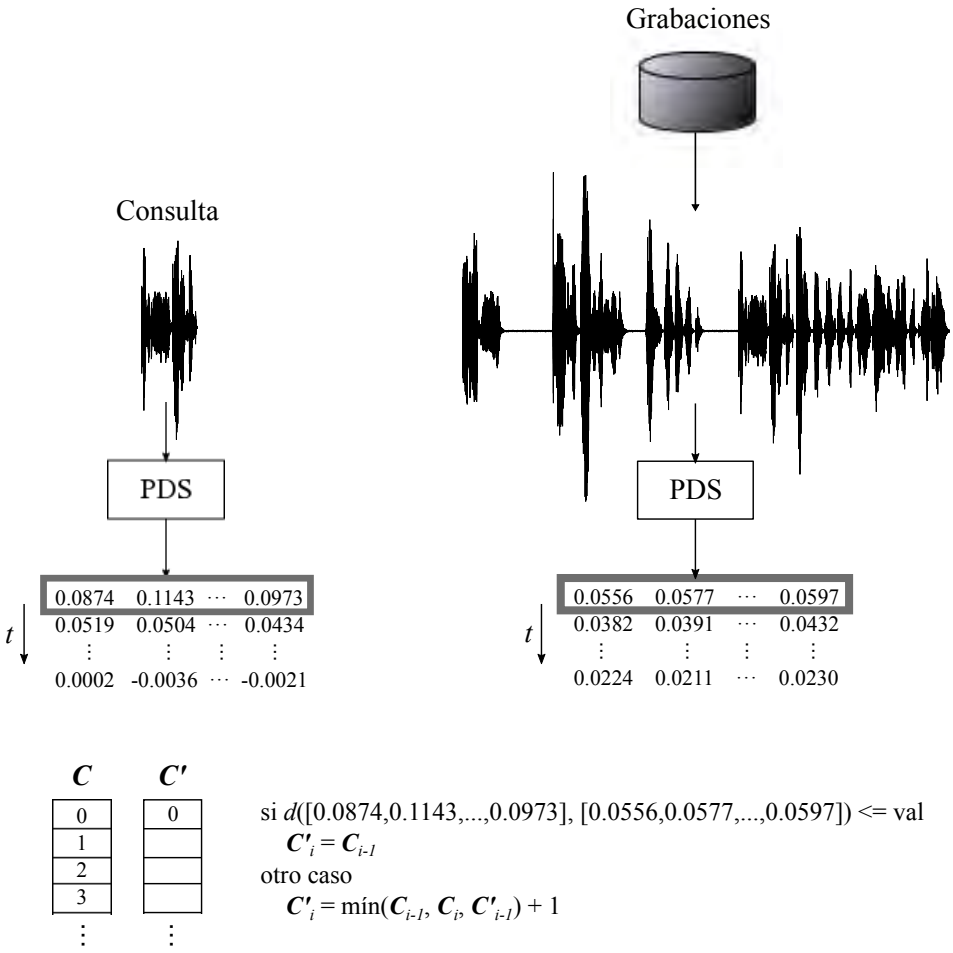


Figura 3.1: Inicio del proceso de búsqueda de una elocución en una base de datos de grabaciones de audio.

En el proceso de búsqueda el primer paso es la captura de la palabra o frase que quiere ser encontrada, en la Figura 3.1 esta elocución está etiquetada como consulta. Una vez que ésta se almacena en memoria se procede a obtener la matriz de características de dicha elocución, tal y como se puede apreciar en la Figura 3.1. El modulo de PDS genera la matriz de características de la elocución y las grabaciones, dicha matriz está denotada por $MC_{t,m}$, $t = 0, 1, \ldots, T$ donde T es el número de marcos y $m = 0, 1, \ldots, M$ donde M es el número de coeficientes del banco de filtros de Mel.

Ya que se ha caracterizado la señal de consulta, se alineará la matriz de características de la consulta con cada una de las matrices de características de los archivos de audio. Camarena-Ibarrola y Chávez en [Camarena-Ibarrola y Chávez, 2006] presentan un esquema de alineamiento de interpretaciones musicales donde en lugar de alinear matrices de características conformadas por los MFCC se alinean dos matrices que conforman una firma de audio (AFP del Inglés *Audio-Fingerprint*). En la Figura 3.2 se ilustra el inicio del alineamiento de dos matrices, donde la matriz MC es la matriz de la consulta y $MC^{'}$ es la n-ésima matriz de la base de datos. Tal y como se puede observar la primera columna acumula la distancia entre un símbolo nulo y una cadena de tamaño i, donde i es el índice que avanza en el tiempo. La columna únicamente almacena una secuencia consecutiva del 0 hasta el número de marcos de la elocución o consulta, i.e., la distancia entre dos símbolos nulos es 0, la distancia entre un símbolo nulo y una cadena de tamaño 1 es igual a 1 y así sucesivamente. Posteriormente cada primer elemento de la columna que se está llenando actualmente ($C^{'}$) es puesto a 0. A partir del segundo elemento se compararan todos los vectores de cada renglón de la matriz MC con el i-ésimo de $MC^{'}$ usando Levenshtein, LCS o twLCS. Una vez realizado lo anterior, se avanza en el tiempo, i.e., se incrementa i y se repite todo el proceso anterior iterativamente hasta que se hayan comparado todos los vectores de MC con los vectores de $MC^{'}$.

Hemos descrito el proceso de alineamiento entre dos matrices de características, donde cada una de dichas matrices no solo caracterizan a cada señal para que se pueda llevar a cabo la identificación de similitudes, sino que reducen la dimensionalidad para que dicho proceso sea realizado rápidamente y estos sistemas puedan ser utilizados en la practica. Sin embargo en un entorno real las grabaciones pueden llegar a ser muy extensas, i.e., pueden

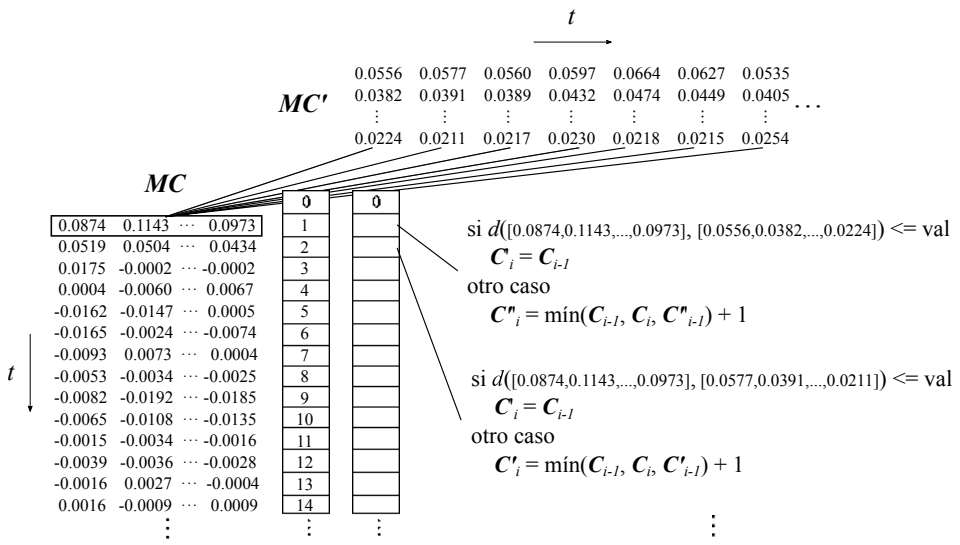


Figura 3.2: Configuración inicial del alineamiento de dos matrices de características.

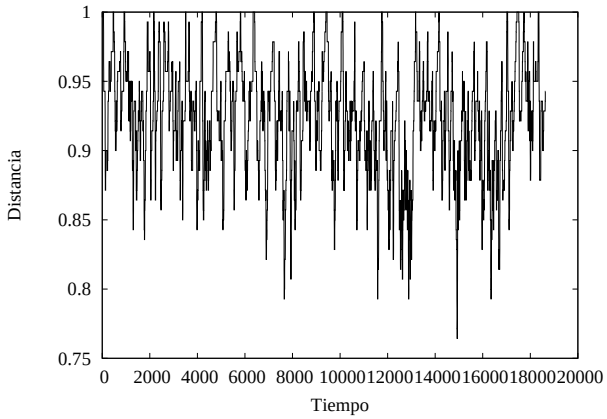
estar compuestas por horas de audio. Si pensamos en una grabación de 2 horas muestreada a 8 kHz, tendríamos una grabación con $57,600,000$ muestras. Como procesamos dichas muestras en marcos de 30 ms cada 10 ms, se tendría en memoria una matriz de $720,000 \times 15$ valores de punto flotante, i.e., podríamos almacenar dicha matriz en 50 MB. Imaginemos que se están ejecutando cinco programas como los que acabamos de mencionar y cada uno está trabajando con archivos de audio y video, puede llegar a ser importante un sistema de búsqueda que ahorre almacenamiento en memoria. Si tomamos en cuenta que el tipo de alineamiento que aplicamos no implica que la matriz de la grabación de referencia tenga que estar totalmente cargada en memoria, podemos cargarla por secciones. Estas secciones no tienen que ser muy pequeñas ya que si la matriz está guardada en un archivo de texto el procedimiento de búsqueda se retardaría debido a las interrupciones de entrada salida que implicaría estar leyendo cada valor de cada vector continuamente.

Entendiendo que en lugar de caracteres se van a alinear secuencias de vectores de características, ¿Cómo vamos a decidir cuando dos vectores (compuestos por los MFCC) son iguales o no?. Dado que el espacio vectorial $\mathbb{R}^D$ al que pertenece cada vector, es también un espacio métrico con la distancia coseno d (definida en (3.1)) como métrica o función de distancia. La respuesta al cuestionamiento es la siguiente. Debido a las características

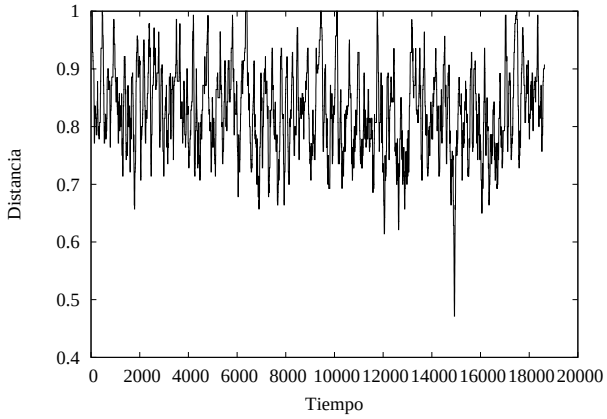
inherentes del habla humana, la probabilidad de que ocurran dos vectores iguales es casi nula. En cambio podemos determinar cuando dos vectores son muy cercanos o parecidos, i.e., cuando la distancia entre ellos es muy pequeña y cuando son muy diferentes si la distancia entre ellos es elevada con respecto al criterio de cercanía que hayamos definido. Este criterio de cercanía consiste en definir un valor umbral, de manera que si la distancia entre dos vectores es menor o igual a dicho umbral se considera que los dos vectores son tan cercanos que se consideran iguales y si la distancia es mayor que el umbral los vectores son diferentes.

$$d = 1 - \left| \frac{\sum_{i=0}^{D-1} a_i b_i}{\sqrt{\sum_{i=0}^{D-1} a_i^2} \sqrt{\sum_{i=0}^{D-1} b_i^2}} \right| \tag{3.1}$$

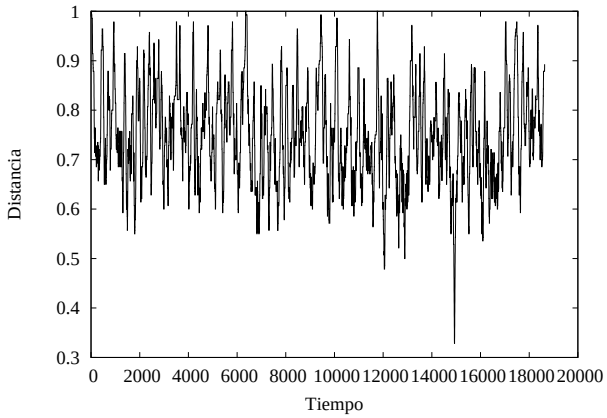
Los valores de cada extremo del rango de distancias obtenido en una simulación, donde se calcula la distancia de todos contra todos los vectores, son de 0.0 a 1.0. El conjunto de vectores fue extraído de una grabación con una duración de 2 min y 35 segundos. Durante la fase de prueba del sistema diseñado se observó que el rango de distancias entre vectores de una elocución y una grabación de voz va desde distancias con valores del orden de las milésimas 0.001 y hasta distancias muy cercanas a 0.99. Por lo tanto, se observaron las distancias obtenidas al llevar a cabo pruebas de alineamiento y se determinó que un valor umbral entre 0.005 y 0.05 permite que el sistema pueda identificar cuando un vector de características es parecido a otro, de tal manera que los vectores puedan ser considerados iguales. En la Figura 3.3 se ilustran las gráficas de los valores de distancia obtenidos durante un alineamiento para distintos valores del criterio de cercanía. En las figuras se observa que en los extremos del rango 0.005 - 0.05 el sistema comienza a entregar distancias similares, no así con el resto de los valores, e.g., en la Figura 3.3(b) se puede apreciar claramente el lugar donde ocurre el valor de distancia menor, dicho lugar corresponde con la posición donde se encuentra la palabra o frase buscada. Si se establece un umbral mayor a 0.05 el sistema se vuelve muy permisivo, regresa distancias muy parecidas a lo largo de todo el proceso de alineamiento, i.e., encuentra un gran número de vectores parecidos entre sí. Por el contrario, si el valor umbral se establece por debajo de un valor de 0.005 el sistema no detecta similitudes, por tanto le es imposible determinar el instante donde se ubica la elocución.



(a) ***Valor de cercanía*** 0.007

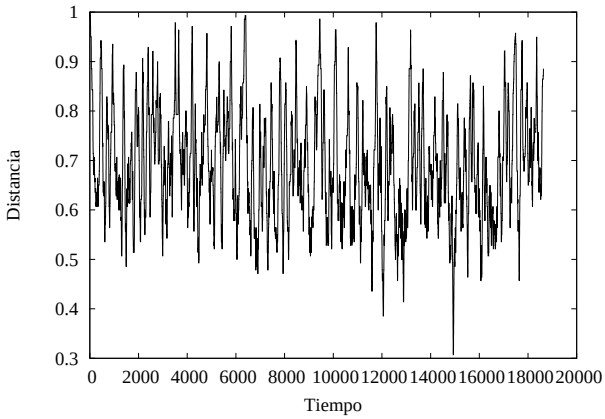


(b) ***Valor de cercanía*** 0.015

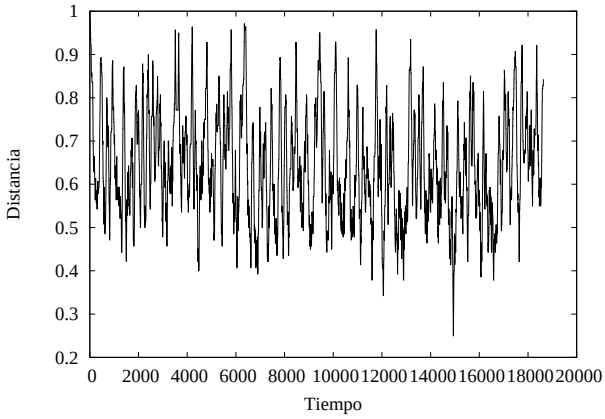


(c) ***Valor de cercanía*** 0.023

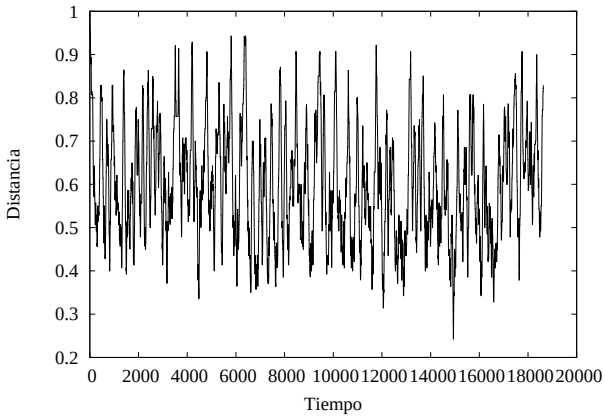
Figura 3.3: Distancias obtenidas a lo largo del alineamiento de una palabra o frase con un flujo de audio.



(d) ***Valor de cercanía*** 0.031



(e) ***Valor de cercanía*** 0.039



(f) ***Valor de cercanía*** 0.047

Figura 3.3: Distancias obtenidas a lo largo del alineamiento de una palabra o frase con un flujo de audio. (*Continuación*)

3.2. Implementación del sistema

3.2.1. Obtención de los MFCC

En este trabajo la caracterización de una señal de audio se realizó por medio de la extracción de los coeficientes cepstrales de Mel (MFCC del Inglés *Mel Frequency Cepstral Coefficients*). Nos basamos en la implementación de Sigurdsson *et al.* en [Sigurdsson et al., 2006] para obtener dichos coeficientes.

La extracción de características se realiza como sigue. Durante el proceso de ventaneo descrito en la Sección 2.2.1, al resultado de la multiplicación del marco por la ventana de Hamming se le aplica la transformada rápida de Fourier (FFT del Ingés *Fast Fourier Transform*) [William H. Press, 1992]. En seguida se obtiene el espectro de magnitud, calculando la magnitud de cada coeficiente obtenido por la FFT, e.g., $\mid c \mid = \mid a + jb \mid = \sqrt{a^2 + b^2}$. Posteriormente el espectro de magnitud es escalado por el banco de filtros de Mel los cuales se obtienen por medio de (3.2),

$$H[k,m] = \begin{cases} 0 & \text{para} \quad f[k] < f_c[m-1] \\ \frac{f[k]-f_c[m-1]}{f_c[m]-f_c[m-1]} & \text{para} \quad f_c[m-1] \leq f[k] < f_c[m] \\ \frac{f[k]-f_c[m+1]}{f_c[m]-f_c[m+1]} & \text{para} \quad f_c[m] \leq f[k] < f_c[m+1] \\ 0 & \text{para} \quad f[k] \geq f_c[m+1] \end{cases} \tag{3.2}$$

para $k = 0, 1, \ldots, N-1$ y $m = 1, 2, \ldots, M$, donde $f[k] = \frac{kf_s}{N}$ es la frecuencia correspondiente a la muestra k del espectro, f_s es la frecuencia de muestreo que en nuestro caso es de 8 kHz y N es el tamaño de marco que nosotros fijamos a 256, ya que es el número potencia de dos más cercano a un marco de 240 muestras que equivale a 30 ms de una señal de audio muestreada a 8 kHz, tal y como se mencionó en la Sección 2.2.1. La frecuencia central del filtro que se está calculando actualmente es $f_c[m]$, la frecuencia central anterior al filtro actual es $f_c[m-1]$ y la frecuencia central siguiente al filtro actual es $f_c[m+1]$. $H[k,m]$ almacena la amplitud que corresponde a cada frecuencia $f[k]$ para el m-ésimo filtro de Mel.

En la Figura 2.7 se mostró un ejemplo del banco de filtros triangulares de Mel, estos fueron calculados con (3.2). Si pensamos en el cálculo de cada filtro individualmente,

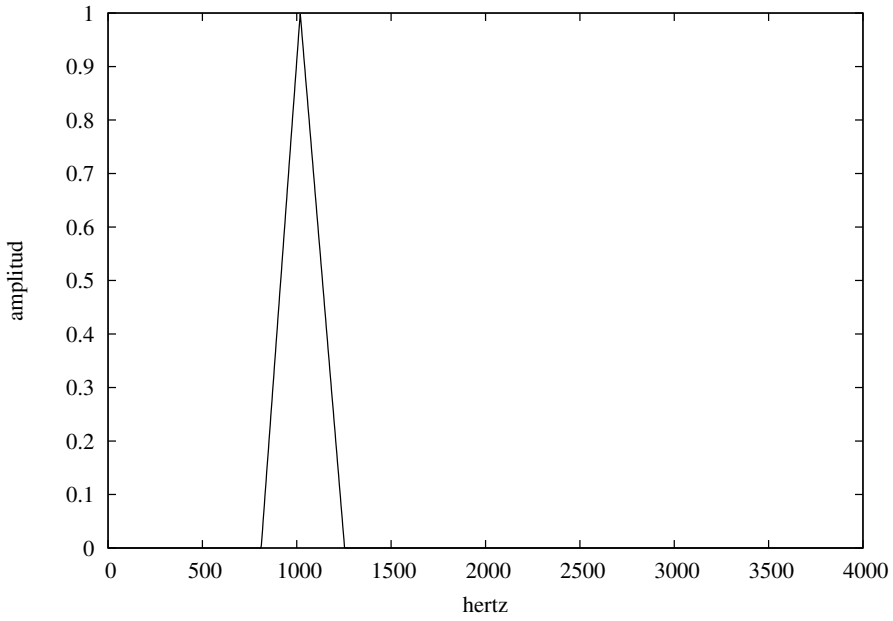


Figura 3.4: Séptimo filtro de Mel del banco de filtros de la Figura 2.7.

para frecuencias menores a $f_c[m-1]$ y mayores a $f_c[m+1]$ es decir frecuencias fuera de la banda del filtro que se está calculando, la amplitud del filtro es igual a 0. A partir de $f_c[m-1]$ y hasta $f_c[m]$ el filtro se calcula con la pendiente $\frac{f[k]-f_c[m-1]}{f_c[m]-f_c[m-1]}$. Para las frecuencias que se ubiquen dentro de $f_c[m]$ y $f_c[m+1]$ los valores del filtro se obtendrán por medio de la pendiente $\frac{f[k]-f_c[m+1]}{f_c[m]-f_c[m+1]}$. En la Figura 3.4 se muestra el séptimo filtro de la Figura 2.7, calculados a partir de la muestra 26 que corresponde con la frecuencia 812.5 y hasta la muestra 40 que corresponde con una frecuencia de 1250.

Al resultado del escalamiento se le aplica el logaritmo, de esta manera habremos escalado tanto en frecuencia como en magnitud. En (3.3) se define el proceso de doble escalado.

$$X^{'}[m] = \ln\left(\sum_{k=0}^{N-1} |X[k]| \cdot H[k,m]\right) \tag{3.3}$$

En la Tabla 3.1 se muestran las equivalencias para el calculo de Mels a través de una frecuencia en hertz y viceversa, para $m = 0, 1, \ldots, M-1$ donde M es el número de filtros de Mel y $\Delta\phi = \frac{(\phi_{max}-\phi_{min})}{(M+1)}$ es el paso fijo con el que se avanza en la escala de Mel

Tabla 3.1: Equivalencia entre hertz y mels.

hertz	mels
$f(frecuencia)$	$\phi = 2595 \log_{10}(\frac{f}{700} + 1)$
$f_c[m] = 700(10^{\frac{\phi_c(m)}{2595}} - 1)$	$\phi_c[m] = m \cdot \Delta\phi$
$f_{max} = \frac{f_s}{2} = 4$ kHz	$\phi_{max} = 2595 \log_{10}(\frac{f_{max}}{700} + 1)$
$f_{min} = 80$ Hz	$\phi_{min} = 2595 \log_{10}(\frac{f_{min}}{700} + 1)$

entre frecuencias centrales de cada filtro triangular.

Por último los MFCC son obtenidos de la aplicación de una de las versiones de la Transformada Discreta Coseno (DCT del Ingés *Discrete Cosine Transform*) a $X^{'}(m)$, como se muestra en (3.4).

$$c[l] = \sum_{m=0}^{M-1} X^{'}[m] \cos\left[l\frac{\pi}{M}(m + 0.5)\right] \quad \text{para} \quad l = 0, 1, \ldots, M-1 \tag{3.4}$$

3.2.2. Obtención de los coeficientes Delta y Delta-Delta

En el Capítulo 2 vimos que agregar las características dinámicas de la voz a los vectores de características obtenidos con determinando esquema de extracción, mejora significativamente el desempeño de un sistema de reconocimiento de voz. Huang *et al.* en [Huang et al., 2001] presentan un esquema de extracción de características que es el estado del arte en reconocimiento de voz, el cual consiste en aumentar un vector de características conformado por 13 coeficientes cepstrales de Mel con sus coeficientes Delta y Delta-Delta (también llamados coeficientes de aceleración). Este esquema que genera vectores de 39 coeficientes (13 MFCC + 13 Deltas + 13 Delta-Deltas), incrementa la reducción del error relativo en un 20 %.

Young *et al.* en [Young et al., 2006] presentan la implementación de los coeficientes Delta y Delta-Delta. En señales de tiempo discreto la derivada es la diferencia entre una muestra y el valor anterior a esa muestra, en (3.5) se muestra una variante de este calculo donde la diferencia se realiza entre MFCC ubicados en los extremos de una ventana de tamaño fijo.

$$\delta_l = \frac{c[l+\theta] - c[l-\theta]}{2\theta} \tag{3.5}$$

Un procedimiento que entrega coeficientes más robustos se muestra en (3.6). El cálculo de los coeficientes delta consiste en sumar las diferencias entre MFCC ubicados en los extremos de una ventana que varía su tamaño de 0 hasta un tamaño máximo predefinido $\Theta - 1$.

$$\delta_l = \frac{\sum_{\theta=0}^{\Theta-1} \theta(c[l+\theta] - c[l-\theta])}{\sum_{\theta=0}^{\Theta-1} \theta^2} \quad \text{para} \quad l = 0, 1, \ldots, M-1 \tag{3.6}$$

Para ambos procedimientos las restricciones al inicio y al final de cada vector de MFCC, se define en (3.7) y (3.8).

$$\delta_l = c[l+1] - c[l], \quad l - \theta < 0 \tag{3.7}$$

$$\delta_l = c[l] - c[l-1], \quad l + \theta > M - 1 \tag{3.8}$$

Una vez que se ha repasado tanto el cálculo del banco de filtros, el proceso de doble escalado, la aplicación de la DCT para reducir la dimensionalidad de los vectores de características y el proceso de hacerlos más robustos aumentándolos con los coeficientes Delta y los coeficientes de aceleración.

El Algoritmo 1 se presenta con el fin de esclarecer la secuencia de cada paso en el cálculo de la matriz de características $MC_{t,m}$, $t = 0, 1, \ldots, T$, $m = 0, 1, \ldots, M$.

En las líneas $2-4$ se realiza la declaración de las variables como son el tamaño de la señal de audio, el número y tamaño de los marcos en los que se va a procesar la señal, el tamaño de ventana para el cálculo de los coeficientes delta y la ventana de Hamming con la que se ponderará al marco que se esté procesando. La sección principal del Algoritmo 1 corresponde con el ciclo de la línea 5, este ciclo es el que realiza el ventaneo de la señal. Cada marco t es copiado temporalmente a un arreglo (cada marco tiene un tamaño de 256 muestras), en seguida se realiza el producto punto del marco con la ventana de Hamming h creada previamente. A continuación se obtienen los MFCC tal y como se mencionó anteriormente en esta sección y por último cada coeficiente es copiado en la m-ésima columna

Algoritmo 1 Extrae la Matriz de Características.

Entrada: *señal*: Señal de voz

Salida: MC: Matriz de características

1: **función** ExtraeMatrizDeCaracteristicas(*señal*)
2: N: Tamaño de la señal, TM: Tamaño de marco
3: $T \leftarrow \frac{N}{80} - 2$, $\Theta \leftarrow 2$
4: $h \leftarrow$CreaVentanaDeHamming( )
5: **para** $paso \leftarrow 0$, $t \leftarrow 0$ **hasta** T **hacer**
6: Copia a w TM muestras de *senial* a partir de *señal*[*paso*]
7: $x \leftarrow w \cdot h$ ▷ Se realiza el producto punto entre el t-ésimo marco y h
8: $X[k] \leftarrow \mathfrak{F}[x[n]]$ ▷ Se calcula la Transformada de Fourier de $x[n]$
9: $X'[m] \leftarrow \ln\left(\sum_{k=0}^{N-1} |X[k]| \cdot H[k,m]\right)$ para $m = 0, 1, \ldots, M-1$
10: $c[l] \leftarrow \mathfrak{C}\left[X'[m]\right]$ ▷ Se calcula la Transformada Coseno de $X'[m]$
11: $\delta[l] \leftarrow \frac{\sum_{\theta=0}^{\Theta-1} \theta(c[l+\theta]-c[l-\theta])}{\sum_{\theta=0}^{\Theta-1} \theta^2}$ para $l = 0, 1, \ldots, M-1$
12: $\delta^2(l) \leftarrow \frac{\sum_{\theta=0}^{\Theta-1} \theta(\delta[l+\theta]-\delta[l-\theta])}{\sum_{\theta=0}^{\Theta-1} \theta^2}$ para $l = 0, 1, \ldots, M-1$
13: $\hat{c} \leftarrow c[l] \cup \delta[l] \cup \delta^2[l]$
14: **para** $m \leftarrow 0$ **hasta** $3 * M$ **hacer**
15: $MC_{t,m} \leftarrow \hat{c}[m]$
16: $paso \leftarrow paso + 80$, $t \leftarrow t + 1$
17: **regresa** MC

de $MC_{t,m}$. Este proceso se repite hasta que se obtengan los MFCC del último marco. Al final se regresa MC que es una caracterización comprimida de la señal de audio completa y que se utilizará en la búsqueda.

3.2.3. Proceso de búsqueda en audio

En la sección anterior (Sección 3.1) se describió cómo se realiza el alineamiento entre matrices de características, la implementación de dicho proceso se muestra en el Algoritmo 2, el cual se describe a continuación. En la declaración de variables la columna C en el caso de la Distancia de Edición, LCS, twLCS y Levenshtein se inicializa con la secuencia consecutiva de distancias entre el caracter nulo y las cadenas de tamaño i (representa la variable del tiempo para MC en esta sección del Algoritmo). Para el caso de el DTW se inicializa con la distancia entre el primer vector de la matriz $MC^{'}$ y el i-ésimo vector de la matriz MC. A continuación en la línea 8 el ciclo principal recorre uno por uno los vectores de la matriz de la referencia $MC^{'}$, en cada iteración se realiza lo siguiente: 1) Se reinicia $C^{'}$, 2) Se llama a la función que lleva a cabo el alineamiento, i.e., el procedimiento TecnicaDeAlineamiento es reemplazado por alguno de los siguientes procedimientos DistanciaDeEdicion, LCS, twLCS, Levenshtein o DTW con los parámetros $MC, MC^{'}_{t}, C, C^{'}$ que ya han sido descritos anteriormente. 3) El último valor de $C^{'}$ que es el que nos indica el parecido de las matrices en ese instante es comparado con un valor umbral. Si el valor en $C^{'}_{N+1}$ es menor que dicho valor umbral, entonces ese marco se considera como una ocurrencia y se almacena su posición i en una lista. 4) Por último se copian los valores de $C^{'}$ a C y se itera hasta que t alcance un valor igual al número de renglones de la matriz $MC^{'}$.

En los Algoritmos 3, 4, 5, 6 y 7, se muestran las implementaciones de los procedimientos que son llamados en la línea 10. Estos cada vez que son llamados reciben la matriz MC completa y solo un vector de $MC^{'}$ para llevar a cabo las comparaciones entre todos los vectores de MC y el vector de $MC^{'}$ que se recibió como parámetro en dicho procedimiento. En los Algoritmos 6 y 7, ya que la distancia coseno entrega distancias entre 0 y 1, la distancia entre los dos vectores se pondera con un valor de 2 para que el costo de la sustitución valga 2 cuando los vectores sean diferentes y cero en caso contrario.

Algoritmo 2 Regresa una lista con las posiciones donde hubo una ocurrencia.

Entrada: MC: Matriz de características de la consulta, $MC^{'}$: Matriz de características de la referencia, *umbral*

Salida: $poss_{min}$: Lista de posiciones de ocurrencias

1: **función** BUSCA(MC, $MC^{'}$)
2: $N \leftarrow MC$.NUMERODERENGLONES( ) ▷ Regresa el número de renglones de MC
3: $T \leftarrow MC^{'}$.NUMERODERENGLONES( ) ▷ Regresa el número de renglones de $MC^{'}$
4: $poss_{min} \leftarrow \{\}$ ▷ Se inicializa la lista como vacía
5: $C \leftarrow$ CEROS(N+1), $C^{'} \leftarrow$ CEROS(N+1) ▷ Se inicializan las columnas a ceros
6: **para** $i \leftarrow 0$ **hasta** $N + 1$ **hacer**
7: $C_i \leftarrow i$
8: **para** $t \leftarrow 0$ **hasta** T **hacer**
9: $C^{'} \leftarrow$ CEROS(N+1)
10: TECNICADEALINEAMIENTO($MC, MC^{'}_t, C, C^{'}$)
11: **si** $C^{'}_{N+1} < umbral$ **entonces**
12: $poss_{min}$.AGREGAPOSICION(t)
13: Copia a C $N + 1$ muestras de $C^{'}$ a partir de $C^{'}_0$
14: **regresa** $poss_{min}$

Algoritmo 3 Procedimiento para el cálculo de la distancia de edición entre MC y el vector de $MC^{'}$ en un determinado instante.

Entrada: MC de la consulta, V: vector de la referencia en un determinado instante, C, $C^{'}$

1: **procedimiento** DISTANCIADEEDICION($senial, MC, V, C, C^{'}$)
2: $D \leftarrow V$.NUMERODECOLUMNAS( ) ▷ Regresa el número de columnas de V
3: **para** $i \leftarrow 1$ **hasta** $C^{'}$.NUMERODERENGLONES( ) **hacer**
4: $d \leftarrow 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} V_j}{\sqrt{\sum_{j=0}^{D-1} MC_{i-1,j}^2} \sqrt{\sum_{j=0}^{D-1} V_j^2}} \right|$
5: **si** $d \leq$ criterio de cercanía **entonces**
6: $C_i^{'} \leftarrow C_{i-1}$
7: **otro caso**
8: $C_i^{'} \leftarrow \text{mín}\left(C_{i-1}^{'}, C_i, C_{i-1}\right) + 1$

Algoritmo 4 Procedimiento para el cálculo de LCS entre MC y el vector de $MC^{'}$ en un determinado instante.

1: **procedimiento** LCS($senial, MC, V, C, C^{'}$)
2: $D \leftarrow V$.NUMERODECOLUMNAS( ) ▷ Regresa el número de columnas de V
3: **para** $i \leftarrow 1$ **hasta** $C^{'}$.NUMERODERENGLONES( ) **hacer**
4: $d \leftarrow 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} V_j}{\sqrt{\sum_{j=0}^{D-1} MC_{i-1,j}^2} \sqrt{\sum_{j=0}^{D-1} V_j^2}} \right|$
5: **si** $d \leq$ criterio de cercanía **entonces**
6: $C_i^{'} \leftarrow C_{i-1}$
7: **otro caso**
8: $C_i^{'} \leftarrow \text{mín}\left(C_{i-1}^{'}, C_i\right) + 1$

Algoritmo 5 Procedimiento para el cálculo de twLCS entre MC y el vector de $MC^{'}$ en un determinado instante.

procedimiento TWLCS($senial, MC, V, C, C^{'}$)
  $D \leftarrow V$.NUMERODECOLUMNAS( ) ▷ Regresa el número de columnas de V
  para $i \leftarrow 1$ **hasta** $C^{'}$.NUMERODERENGLONES( ) **hacer**
    $d \leftarrow 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} V_j}{\sqrt{\sum_{j=0}^{D-1} MC_{i-1,j}^2} \sqrt{\sum_{j=0}^{D-1} V_j^2}} \right|$
    si $d \leq$ criterio de cercanía **entonces**
      $C_i^{'} \leftarrow \text{mín}\left(C_{i-1}, C_i, C_{i-1}^{'}\right)$
    otro caso
      $C_i^{'} \leftarrow \text{mín}\left(C_{i-1}^{'}, C_i\right) + 1$

Algoritmo 6 Procedimiento para el cálculo de Levenshtein entre MC y el vector de $MC^{'}$ en un determinado instante, usando $2d$ como costo de la operación de sustitución.

procedimiento LEVENSHTEIN($senial, MC, V, C, C^{'}$)
  $D \leftarrow V$.NUMERODECOLUMNAS( ) ▷ Regresa el número de columnas de V
  para $i \leftarrow 1$ **hasta** $C^{'}$.NUMERODERENGLONES( ) **hacer**
    $d \leftarrow 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} V_j}{\sqrt{\sum_{j=0}^{D-1} MC_{i-1,j}^2} \sqrt{\sum_{j=0}^{D-1} V_j^2}} \right|$
    $C_i^{'} \leftarrow \text{mín}\left(C_{i-1} + 2d, C_i + 1, C_{i-1}^{'} + 1\right)$

Algoritmo 7 Procedimiento para el cálculo de DTW entre MC y el vector de $MC^{'}$ en un determinado instante.

procedimiento DTW($senial, MC, V, C, C^{'}$)
  $D \leftarrow V$.NUMERODECOLUMNAS( ) ▷ Regresa el número de columnas de V
  para $i \leftarrow 1$ **hasta** $C^{'}$.NUMERODERENGLONES( ) **hacer**
    $d \leftarrow 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} V_j}{\sqrt{\sum_{j=0}^{D-1} MC_{i-1,j}^2} \sqrt{\sum_{j=0}^{D-1} V_j^2}} \right|$
    $C_i^{'} \leftarrow \text{mín}\left(C_{i-1} + 2d, C_i + d, C_{i-1}^{'} + d\right)$

3.3. Conclusiones del Capítulo

El alineamiento entre matrices de características se lleva a cabo comparando cada vector de la consulta con cada uno de los vectores de la matriz del archivo de referencia, al final el último valor de la columna que se está llenando es la distancia en ese instante de tiempo.

Para la mayoría de los esquemas de ASM utilizados en este trabajo, es necesario establecer un valor de cercanía. Esto con el fin de poder llevar a cabo un alineamiento de matrices de características correcto. Dicha variable puede tomar valores entre 0.005 y 0.05 aproximadamente. Este rango se determinó experimentalmente observando las gráficas de distancias obtenidas a lo largo de un alineamiento.

El DTW y la variante de la distancia de Levenshtein son alternativas donde no es necesario hacer uso de un criterio de cercanía para decidir que operación de edición se va a realizar.

El proceso de búsqueda monitorea las distancias obtenidas al llevar acabo el alineamiento y si encuentra que una distancia es menor que un valor umbral predefinido, registra que en ese instante ocurrió la consulta y almacena la posición de dicho marco. Al final el algoritmo regresa una lista con las posiciones donde ocurrió la consulta.

En el capítulo 4, que es el capítulo que se presenta a continuación, se llevó a cabo la experimentación para determinar cual de las técnicas implementadas hace que el sistema tenga un mejor desempeño.

Capítulo 4

Experimentos y Resultados

En este capítulo se presentan en primera instancia los elementos que fueron usados para poder llevar a cabo los experimentos, que van desde las características del equipo de computo utilizado, aplicaciones que se adaptaron para grabado y reproducción, así como las elocuciones de referencia y de consulta que fueron creadas. En segunda instancia se presentan los experimentos realizados de los cuales se obtuvieron curvas ROC para poder evaluar el desempeño del sistema. En tercera instancia se presenta la evaluación del sistema utilizando consultas grabadas por diferentes hablantes. Finalmente se presentan los tiempos de ejecución promedio obtenidos al llevar a cabo los experimentos de búsqueda.

4.1. Entorno de experimentación

El equipo con el que se trabajó durante el desarrollo de este trabajo es una computadora integrada por un procesador con un tamaño de instrucciones de 64 bits, con dos núcleos y cuatro subprocesos a una frecuencia de 2.9 GHz y frecuencia turbo máxima de 3.6 GHz. También cuenta con una memoria principal de 8 GB.

Se desarrollaron dos aplicaciones auxiliares para llevar a cabo la experimentación. La primera aplicación es una adaptación del demo de la API Java Sound de Java^{TM}, esta aplicación se usó para realizar grabaciones y poder extraer características de las mismas. Las grabaciones realizadas se dividen en dos conjuntos, las grabaciones de referencia, que son donde se llevaron a cabo las búsquedas y las grabaciones de elocuciones de palabras

y frases que fueron elegidas de las grabaciones de referencia y conforman el conjunto de consultas.

La segunda aplicación consiste en otra adaptación del mismo demo. En este caso, la aplicación reproduce grabaciones de audio a partir de determinadas muestras o instantes de tiempo que son leídos de un archivo de texto plano. Dicho archivo es creado por el sistema de búsqueda que se desarrolló en este trabajo. Esta aplicación fue usada para verificar el funcionamiento del sistema de búsqueda, una vez que se comprobó que el sistema funcionaba correctamente, i.e., que era capaz de ubicar las elocuciones de palabras o frases en una grabación, la aplicación en cuestión fue utilizada para ubicar los instantes en los que ocurren las elocuciones de consulta en las grabaciones de referencia y así poder automatizar los experimentos que se lleváron a cabo.

El marco de trabajo completo con el que se experimentó, involucra al sistema de búsqueda y adicionalmente a las dos aplicaciones que se acaban de mencionar. La utilización de cada aplicación lleva una secuencia lógica. Primero se necesitan realizar las grabaciones en las que se va a buscar y posteriormente seleccionar extractos de esas grabaciones que corresponden con elocuciones de palabras y frases que serán utilizadas como consultas. Este proceso corresponde con la etapa de grabación. En seguida una vez que se cuenta con los conjuntos de referencia y consulta, se pueden llevar a cabo los experimentos de búsqueda de palabras o frases. Por último para validar el comportamiento del sistema se llevó a cabo el cálculo de Curvas ROC (del Inglés *Receiver Operating Characteristic*).

4.2. Grabación y extracción de características

Empecemos por definir los parámetros utilizados en la digitalización de la señal de voz. Todas las grabaciones fueron digitalizadas con una frecuencia de muestreo de 8 kHz, con un tamaño de muestra de 16 bits con signo y las muestras pertenecen a un solo canal.

El primer conjunto de grabaciones que se llevó a cabo, fue el conjunto de grabaciones de referencia, i.e., los flujos de audio en donde se van a llevar a cabo las búsquedas. El procedimiento que se realizó es el siguiente: de un libro digital de texto, en este caso el libro Rayuela de Julio Cortázar, se seleccionaron un grupo de páginas, este grupo consta

de conjuntos de dos páginas consecutivas y cada conjunto corresponde con una grabación. Una vez que creamos las grabaciones de referencia, procedimos a extraer sus características y las matrices obtenidas fueron almacenadas en archivos de texto.

El segundo conjunto de grabaciones consiste en las palabras o frases que van a ser localizadas en las grabaciones creadas en el paso anterior. Primero, se seleccionaron extractos de cada una de las grabaciones que ya se tienen. En total se eligieron 48 extractos, 24 palabras y 24 frases. Una vez que se seleccionaron los 48 extractos que van a ser utilizados como consulta se procedió a la grabación. Cinco personas, dos mujeres y tres hombres, grabaron los 48 extractos. En nuestro caso, que fuimos los encargados de generar el conjunto de grabaciones de referencia, se grabaron dos elocuciones de cada uno de los 48 extractos. La razón de esto es que una elocución de las dos realizadas de cada extracto fue usada para que el sistema de búsqueda la localizara y guardar un registro del instante en el que ocurrió dentro de la grabación. La otra elocución de consulta de ese mismo extracto fue usada en los experimentos de búsqueda que serán presentados en la siguiente sección. De tal manera que si el sistema funciona correctamente, los instantes que regresa el sistema de búsqueda usando de consulta las dos elocuciones de un mismo extracto (validando primero que el instante regresado cuando se busque la primera elocución sea correcto) deben de ser si no iguales (que sería el mejor de los casos para un sistema dependiente del usuario) muy cercanas (que sería el comportamiento esperado en un sistema independiente del usuario).

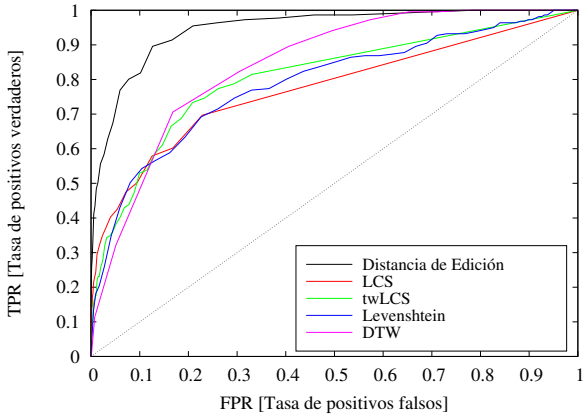
4.3. Experimentos de búsqueda

Los experimentos llevados a cabo consisten en procesar y buscar cada una de las 48 consultas en el conjunto de grabaciones de referencia. Dichas grabaciones de referencia, así como las consultas fueron realizadas por nosotros. Por último los resultados obtenidos de las búsquedas se utilizaron para generar curvas ROC y evaluar el rendimiento del sistema.

En el caso de la Distancia de Edición, LCS y twLCS, donde el sistema tiene como parámetro principal un valor que define un criterio de cercanía, se llevaron a cabo varias repeticiones del experimento planteado en el párrafo anterior para diferentes valores de dicho criterio en el rango de 0.006 a 0.024. De dicho rango se utilizó el primer valor en este

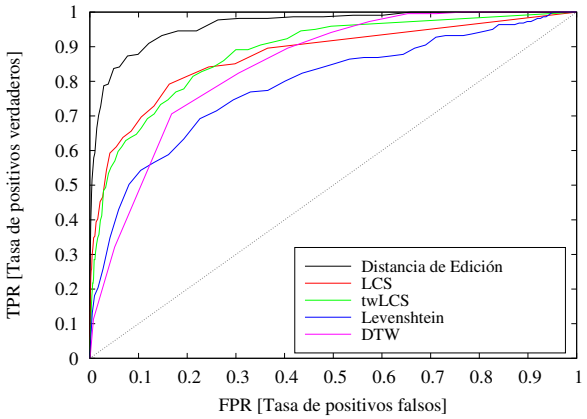
caso 0.006, el segundo valor 0.007 se descarto, el tercer valor 0.008 se utilizó y el cuarto no, etc. El DTW y Levenshtein no dependen de dicho valor por tanto solo se realizó una sola búsqueda de las consultas usando dichas técnicas de alineamiento.

Durante la búsqueda, si la distancia cae por debajo de un valor umbral, se tiene que verificar si esa ocurrencia es verdadera o si hubo un falso positivo. De igual manera si la distancia está por encima de un valor umbral, se tiene que validar que efectivamente la consulta buscada no se encuentre en ese instante, i.e., que ese negativo sea verdadero, de lo contrario habrá ocurrido un falso negativo. Al finalizar la búsqueda de las 48 consultas, para cierto valor umbral se habrán contabilizado el número de Positivos verdaderos TP, Negativos verdaderos TN, Positivos falsos FP y Negativos Falsos FN que haya arrojado el sistema y será posible calcular tanto la tasa de positivos verdaderos (TPR por sus siglas en Inglés True Positive Rate) como la tasa de positivos falsos (FPR por sus siglas en Inglés False Positive Rate), $TPR = TP/(TP + FN)$ y $FPR = FP/(FP + TN)$. Al concluir el experimento se obtuvieron las Curvas ROC graficando TPR contra FPR para todos los valores que puede tomar un valor umbral, en este caso esos valores corresponden con todo el rango dinámico de distancias normalizadas de 0 a 1. Las curvas ROC generadas nos sirvieron para analizar el desempeño de los métodos para diferentes valores del criterio de cercanía. En la Tabla 4.1 se muestran las curvas ROC obtenidas de los experimentos de búsqueda.

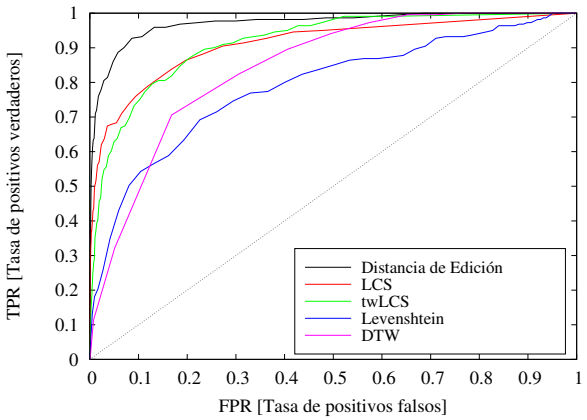


(a) ***Valor de cercanía*** 0.006

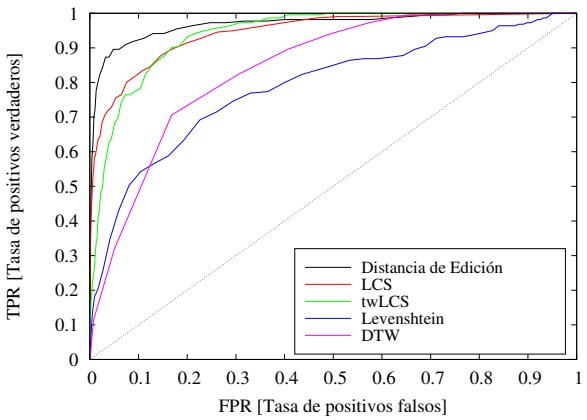
Figura 4.1: Curvas ROC para diferentes valores de cercanía.



(b) *Valor de cercanía* 0.008

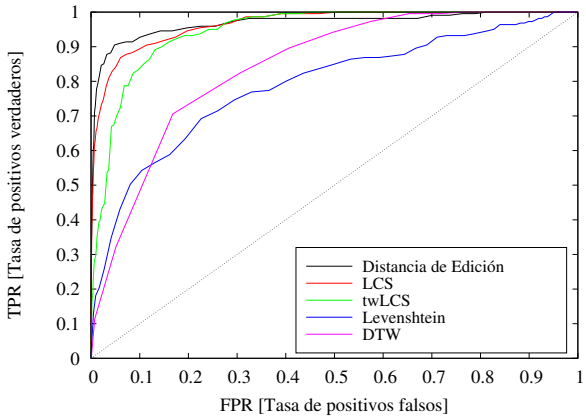


(c) *Valor de cercanía* 0.01

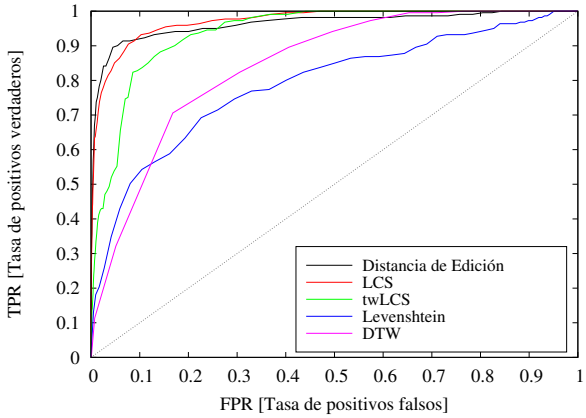


(d) *Valor de cercanía* 0.012

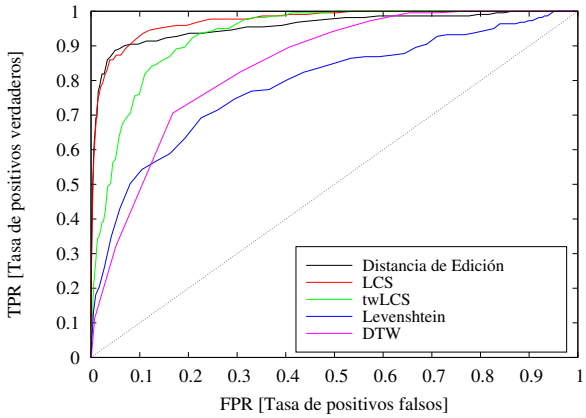
Figura 4.1: Curvas ROC para diferentes valores de cercanía. (*Continuación*)



(e) ***Valor de cercanía*** 0.014

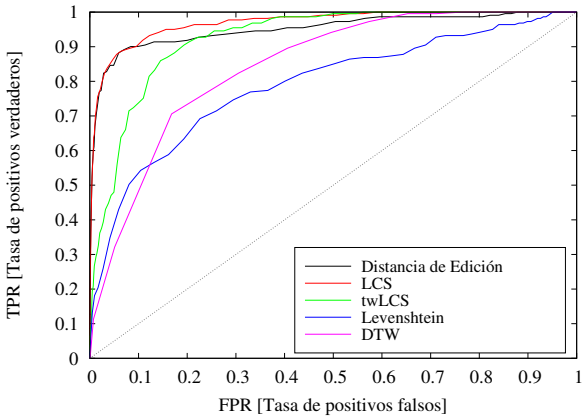


(f) ***Valor de cercanía*** 0.016

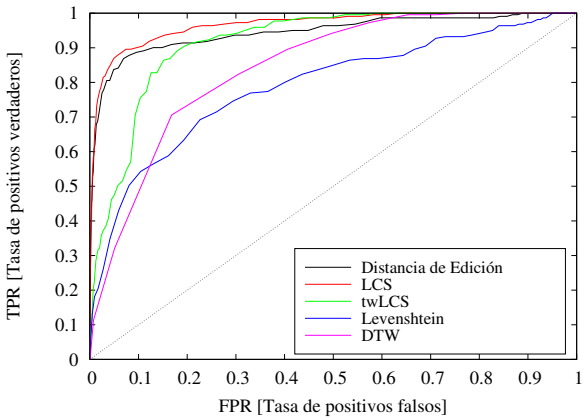


(g) ***Valor de cercanía*** 0.018

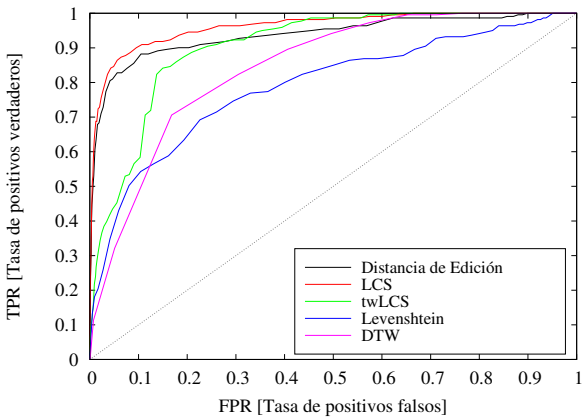
Figura 4.1: Curvas ROC para diferentes valores de cercanía. (*Continuación*)



(h) ***Valor de cercanía*** 0.02

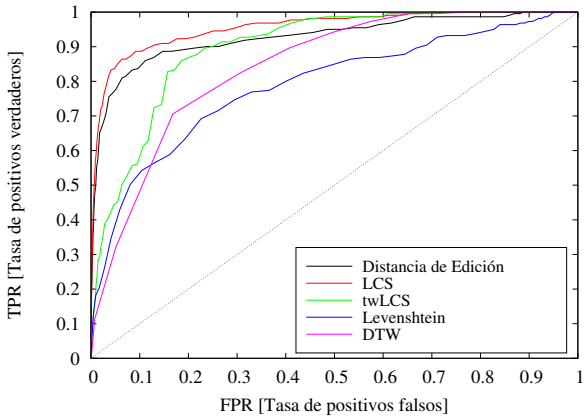


(i) ***Valor de cercanía*** 0.022



(j) ***Valor de cercanía*** 0.024

Figura 4.1: Curvas ROC para diferentes valores de cercanía. (*Continuación*)



(k) ***Valor de cercanía*** 0.026

Figura 4.1: Curvas ROC para diferentes valores de cercanía. (*Continuación*)

4.3.1. Análisis del desempeño de las técnicas de alineamiento mediante Curvas ROC

La Curva ROC, analizada por Fawcett en [Fawcett, 2006], muestra gráficamente la relación de las ocasiones en las que el sistema determina que se ha encontrado la ocurrencia de una consulta en un flujo de audio y que dicha ocurrencia es correcta TPR, contra las ocasiones que el sistema determina que hubo una ocurrencia pero dicha ocurrencia no corresponde con la localización de la consulta dentro del flujo de audio, FPR. TPR corresponde con el eje vertical y FPR con el horizontal, teniendo ambos ejes un rango de 0 a 1. Por tanto un sistema de búsqueda con un funcionamiento óptimo sería aquel que a partir del valor más pequeño que pueda tomar el umbral tiene un valor de TPR de 1 y de FPR de 0, y así para todos los valores que tome el umbral. El sistema que genere puntos de una Curva ROC con la misma contribución tanto de TPR como de FPR, se dice que para ese valor umbral el sistema está haciendo conjeturas al azar. Usaremos el área bajo la Curva ROC (AUC del Inglés *Area Under the ROC Curve*), analizada igualmente por Fawcett en [Fawcett, 2006], para realizar comparaciones del desempeño del sistema usando las diferentes técnicas de alineamiento que se decidieron probar. El AUC de un clasificador es equivalente a la probabilidad de que el clasificador calificará a una instancia positiva elegida aleatoriamente más alto que a una instancia negativa elegida aleatoriamente. Para

obtener el AUC se utilizó la librería de python™*scikit-learn*[1].

Las curvas mostradas en las Figuras 4.1(a) - 4.1(k), confirman que el esquema que se planteó en este trabajo, sí es capaz de localizar elocuciones de palabras o frases en archivos de audio con una tasa de aciertos buena usando cualquiera de las 5 técnicas de alineamiento antes descritas.

La segunda observación que comienza a ser aparente a partir de la Figura 4.1(c), es que los métodos donde se utiliza un valor como criterio de cercanía propician que la tasa de aciertos del sistema aumente con respecto a la tasa que se obtiene con los métodos que no usan este criterio.

Como se mencionó anteriormente el rango de valores del criterio de cercanía que se usó va de 0.006 a 0.026. En el caso de las técnicas de alineamiento que usan dicho criterio, podemos decir que a partir de un valor del criterio de cercanía de 0.006, el sistema es capaz de discriminar cuando dos vectores pueden ser tomados como iguales o no. Para ese valor inicial de dicho criterio las técnicas de alineamiento, a excepción de la distancia de edición que es la que mejor tasa de aciertos le da al sistema, hacen que el sistema tenga una tasa de aciertos buena y con una diferencia de AUC entre la mejor y la peor de ellas de 7 unidades, 0.7729 y 0.8431.

De la Figura 4.1(a) a la 4.1(g) el desempeño del sistema usando la distancia de edición, LCS y twLCS, mejora mientras asciende el valor del criterio de cercanía. Para un valor de 0.018 de dicho criterio y usando LCS el sistema obtiene su máxima tasa de aciertos con una AUC de 0.9723. Con la distancia de edición se obtienen una AUC de 0.9574 y usando twLCS la AUC es de 0.9320. Finalmente con el DTW y Levenshtein se obtienen unas AUC de 0.8431 y 0.7886 respectivamente (ver Figura 4.1(g)). La segunda y tercer mejor tasa de aciertos se obtienen cuando el criterio de cercanía es de 0.016 y se usa LCS obteniendo una AUC de 0.9715 (ver Figura 4.1(f)). En la Figura 4.1(d) el sistema obtiene una AUC de 0.9691 usando la distancia de edición y un criterio de cercanía de 0.012 para realizar la búsqueda.

A partir de un criterio de cercanía mayor a 0.018 el desempeño del sistema comienza a disminuir. Esto se puede interpretar como que a partir de cierto punto mientras

[1]http://scikit-learn.org

dicho criterio aumente el sistema comienza a tomar a dos vectores como parecidos cuando acusticamente no lo son.

4.4. Evaluación del sistema con diferentes hablantes

Para evaluar la precisión del sistema en circunstancias donde las grabaciones de referencia y las de consulta son hechas por diferentes hablantes (sistema independiente del hablante), se llevó a cabo la búsqueda de los 48 extractos grabados por cada una de las cuatro personas que no participaron en la grabación del conjunto de grabaciones de referencia.

Dado que existen diferencias significativas entre elocuciones de una misma palabra o frase pronunciada por personas diferentes, los experimentos de búsqueda que se llevaron a cabo utilizaron un rango más amplio de valores del criterio de cercanía con respecto al que se utilizó en el experimento anterior con el fin de localizar el valor optimo de dicho criterio para otros hablantes, dicho rango se fijo a valores entre 0.002 y 0.030. Al igual que en el experimento anterior se pretendió obtener una curva ROC por cada valor de cercanía y técnica de alineamiento que se utilice. Basándonos en las diferencias acústicas que hay entre la voz de cada parlante se tomó como TP cada instante en el que la distancia cayera por debajo de un valor umbral y que este instante estuviera relativamente cercano a la verdadera ocurrencia, i.e., $pos_{verdadera} - holgura <= pos_{ocurrencia} <= pos_{verdadera} + holgura$, para valores de holgura de 128, 256, 512 y 1024. El resto de parámetros se mantuvo igual con respecto al experimento anterior.

4.4.1. Resultados de la evaluación

Los resultados obtenidos no fueron satisfactorios. Las búsquedas realizadas de las elocuciones grabadas por cada uno de los cuatro hablantes voluntarios no tuvieron éxito, i.e., no se encontró una ocurrencia correcta en alguna de dichas búsquedas.

En las Tablas 4.1(a) - 4.1(d) se muestra la contabilización de FP, TN, TP y FN para cada valor que toma el umbral en el rango dinámico de distancias normalizadas (0.0 a 1.0), usando la distancia de edición como técnica de alineamiento y con una holgura

de 500. Estas gráficas corresponderían con una curva ROC de haber arrojado resultados positivos. En cambio se decidió mostrar el conteo de las variables involucradas en el cálculo de TPR y FPR, ya que no se hubiera podido llevar a cabo el cálculo de TPR ($TPR = TP/(TP + FN)$) dado que a lo largo de los experimentos se obtuvo un conteo de 0 tanto de positivos verdaderos como de falsos negativos.

4.5. Tiempos promedio de ejecución

Se realizaron experimentos para medir el tiempo en el que el sistema termina de realizar una búsqueda de una palabra o frase en un conjunto de grabaciones de referencia.

Los parámetros usados para llevar a cabo las búsquedas son los siguientes:

a) Un valor umbral de 0.2. Para este valor umbral el sistema tiene una tasa de aciertos alta.

Tabla 4.1: Contabilización de FP, TN, TP, FN para cada valor del umbral.

(a) *hablante masculino 1*

umbral	FP	TN	TP	FN
0.0	0	3169296	0	0
0.1	0	3169296	0	0
0.2	38	3169258	0	0
0.3	581	3168715	0	0
0.4	3609	3165687	0	0
0.5	18167	3151129	0	0
0.6	104996	3064300	0	0
0.7	458373	2710923	0	0
0.8	1333355	1835941	0	0
0.9	2379418	789878	0	0
1.0	3054890	114406	0	0

(b) *hablante masculino 2*

umbral	FP	TN	TP	FN
0.0	0	3169296	0	0
0.1	0	3169296	0	0
0.2	17	3169279	0	0
0.3	442	3168854	0	0
0.4	2960	3166336	0	0
0.5	18618	3150678	0	0
0.6	121373	3047923	0	0
0.7	579773	2589523	0	0
0.8	1609123	1560173	0	0
0.9	2567902	601394	0	0
1.0	2993845	175451	0	0

Continúa en la siguiente página

Continuación

Tabla 4.1: Contabilización de FP, TN, TP, FN para cada valor del umbral. (*Continuación*)

(c) *hablante femenino 1*

umbral	FP	TN	TP	FN
0.0	0	3169296	0	0
0.1	13	3169283	0	0
0.2	314	3168982	0	0
0.3	2518	3166778	0	0
0.4	13955	3155341	0	0
0.5	70562	3098734	0	0
0.6	280953	2888343	0	0
0.7	830393	2338903	0	0
0.8	1668091	1501205	0	0
0.9	2443000	726296	0	0
1.0	2862613	306683	0	0

(d) *hablante femenino 2*

umbral	FP	TN	TP	FN
0.0	0	3169296	0	0
0.1	0	3169296	0	0
0.2	444	3168852	0	0
0.3	5985	3163311	0	0
0.4	41492	3127804	0	0
0.5	199355	2969941	0	0
0.6	706602	2462694	0	0
0.7	1577332	1591964	0	0
0.8	2348070	821226	0	0
0.9	2805891	363405	0	0
1.0	3025045	144251	0	0

b) Valores del criterio de cercanía empezando en 0.006 y terminando en 0.026, con incrementos de 0.002 en cada cambió de este valor.

c) Los conjuntos de grabaciones de consulta (24 palabras y 24 frases) y de referencia (11 min de audio) utilizados.

La medición del tiempo de ejecución inicia después de cargar cada grabación que será usada como consulta. La consulta cargada en memoria será buscada en cada grabación de referencia. Una vez que el sistema termine la búsqueda de dicha consulta en la ultima grabación de referencia, se termina la medición. Dado que las grabaciones que se usan como consulta son 48, por cada valor del criterio de cercanía se obtienen 48 mediciones las cuales son promediadas. En la Tabla 4.2 se muestra el tiempo de búsqueda promedio que se obtuvo usando cada una de las cinco técnicas de alineamiento y para cada valor de cercanía.

De la Tabla 4.2 podemos observar que el algoritmo de búsqueda que se desarrolló es capaz de obtener una respuesta en aproximadamente un segundo cuando se busca una

Tabla 4.2: Tiempos promedio de búsqueda. TA: Técnica de Alineamiento; CC: Criterio de Cercanía.

CC \ TA	Distancia de Edición	LCS	twLCS	Levenshtein	DTW
0.006	0.8425	0.8400	0.8347	0.8677	1.5780
0.008	0.8358	0.8352	0.8334	0.8572	1.5705
0.01	0.8390	0.8367	0.8376	0.8600	1.5767
0.012	0.8627	0.8362	0.8375	0.8607	1.5755
0.014	0.8369	0.8373	0.8374	0.8602	1.5736
0.016	0.8376	0.8367	0.8395	0.8603	1.5733
0.018	0.8370	0.8376	0.8380	0.8604	1.5738
0.02	0.8361	0.8352	0.8413	0.8605	1.5747
0.022	0.8399	0.8375	0.8410	0.8571	1.5741
0.024	0.8410	0.8354	0.8435	0.8610	1.5730
0.026	0.8392	0.8381	0.8410	0.8575	1.5738
Promedio	0.8407	0.8369	0.8386	0.8602	1.5743

palabra o frase en 11 min de audio. El sistema obtiene el menor tiempo promedio si usa LCS como técnica de alineamiento. Como se define en (4.2) esto se debe a que esta técnica únicamente permite realizar dos operaciones de edición. Si los vectores son diferentes, LCS calcula el menor de dos valores. Usando la twLCS, el sistema obtiene tiempos de ejecución cercanos a los que se obtienen con LCS. La distancia de edición a diferencia de twLCS permite tres operaciones de edición, i.e., calcula el menor de tres valores cuando los vectores son diferentes, tal y como se muestra en (4.1). twLCS como se indica en (4.3), calcula el mínimo de dos valores sean iguales los dos vectores o no, sin embargo el hecho de realizar dos comparaciones acelera el proceso de búsqueda.

Distancia de Edición

$$d = 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} MC'_{t,j}}{\sqrt{\sum_{j=0}^{D-1} (MC_{i-1,j})^2} \sqrt{\sum_{j=0}^{D-1} (MC'_{t,j})^2}} \right|$$

$$C'_i = \begin{cases} C_{i-1} & \text{si } d <= cc \\ \text{mín}\left(C_{i-1}, C'_{i-1}, C_i\right) + 1 & \text{de otro modo} \end{cases} \tag{4.1}$$

para $i = 1, 2, \ldots, ren(MC)$ y $t = 1, 2, \ldots, ren(MC')$.

LCS

$$d = 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} MC'_{t,j}}{\sqrt{\sum_{j=0}^{D-1} (MC_{i-1,j})^2} \sqrt{\sum_{j=0}^{D-1} (MC'_{t,j})^2}} \right|$$

$$C'_i = \begin{cases} C_{i-1} & \text{si } d <= cc \\ \text{mín}\left(C'_{i-1}, C_i\right) + 1 & \text{de otro modo} \end{cases} \tag{4.2}$$

para $i = 1, 2, \ldots, ren(MC)$ y $t = 1, 2, \ldots, ren(MC')$.

twLCS

$$d = 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} MC'_{t,j}}{\sqrt{\sum_{j=0}^{D-1} (MC_{i-1,j})^2} \sqrt{\sum_{j=0}^{D-1} (MC'_{t,j})^2}} \right|$$

$$C'_i = \begin{cases} \text{mín}\left(C_i, C_{i-1}, C'_{i-1}\right) & \text{si } d <= cc \\ \text{mín}\left(C_i, C'_{i-1}\right) + 1 & \text{de otro modo} \end{cases} \tag{4.3}$$

para $i = 1, 2, \ldots, ren(MC)$ y $t = 1, 2, \ldots, ren(MC')$.

Si el sistema de búsqueda usa como técnicas de alineamiento Levenshtein o DTW, obtiene tiempos de ejecución mayores que los tres métodos anteriores. Esto se debe a que Levenshtein y DTW buscan el mínimo de tres valores en cada instante del alineamiento como se muestra en (4.4) y (4.5).

Levenshtein

$$d = 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} MC'_{t,j}}{\sqrt{\sum_{j=0}^{D-1} (MC_{i-1,j})^2} \sqrt{\sum_{j=0}^{D-1} (MC'_{t,j})^2}} \right|$$

$$C'_i = \text{mín} \begin{cases} C_{i-1} + 2d_{i,j} \\ C'_{i-1} + 1 \\ C_i + 1 \end{cases} \tag{4.4}$$

para $i = 1, 2, \ldots, ren(MC)$ y $t = 1, 2, \ldots, ren(MC')$.

DTW

$$d = 1 - \left| \frac{\sum_{j=0}^{D-1} MC_{i-1,j} MC'_{t,j}}{\sqrt{\sum_{j=0}^{D-1} (MC_{i-1,j})^2} \sqrt{\sum_{j=0}^{D-1} (MC'_{t,j})^2}} \right|$$

$$C'_i = \text{mín} \begin{cases} C_{i-1} + 2d_{i,j} \\ C'_{i-1} + d_{i,j} \\ C_i + d_{i,j} \end{cases} \tag{4.5}$$

para $i = 1, 2, \ldots, ren(MC)$ y $t = 1, 2, \ldots, ren(MC')$.

4.6. Conclusiones del Capítulo

El uso de técnicas de ASM en el problema de la búsqueda de palabras o frases en flujos de audio es muy eficiente. Para un valor del criterio de cercanía igual o mayor a 0.012 los modelos presentan pocos cruces, i.e., se puede decir que un sistema de búsqueda como el que se implementó presenta un mejor desempeño usando LCS y como una buena alternativa la Distancia de Edición. twLCS a pesar de proporcionar al sistema un desempeño por debajo de los dos métodos anteriores, es un buen modelo. DTW y Levenshtein que son los métodos que no dependen de un valor de cercanía, presentan desempeños buenos pero por debajo de los que sí dependen de este valor.

Los resultados obtenidos en la valoración del funcionamiento del sistema de búsqueda como un sistema independiente del usuario arrojaron resultados que no fueron satisfacto-

rios y por tanto establecen que el sistema no admite la búsqueda en grabaciones de referencia si las elocuciones usadas para consultar no son grabadas por el mismo hablante.

Dado que el sistema da una respuesta de aproximadamente un segundo cuando se busca en 11 min de audio, usando cualquiera de las 5 técnicas, es posible que se puedan llevar a cabo sistemas de monitoreo de conversaciones en línea. Ya que el sistema realizaría el alineamiento casi al instante de que se capture cada palabra enunciada por el individuo que esté siendo monitoreado.

Capítulo 5

Conclusiones y Trabajos Futuros

5.1. Conclusiones

El uso de técnicas de ASM, recuperación de música y reconocimiento de voz junto con la obtención de características robustas de la señal de voz como son los MFCC, los coeficientes Delta y Delta-Delta (coeficientes de aceleración), resultan una alternativa eficiente para la búsqueda de palabras o frases en archivos de audio siempre y cuando tanto las grabaciones en las que se busque como las que se usarán para consultar sean hechas por el mismo hablante.

Para el caso de la Distancia de Edición, LCS y twLCS, las Curvas ROC obtenidas muestran que el sistema funciona muy bien a pesar de que depende de un valor de cercanía. El sistema tiene que moverse en el rango que se determino experimentalmente para trabajar correctamente y poder diferenciar cuando dos vectores son lo suficientemente parecidos para tomarlos como iguales. Los métodos DTW y Levenshtein a pesar de no tener el mismo desempeño que los métodos anteriores, le dan un desempeño aceptable al sistema.

5.1.1. Conclusiones particulares

- Aumentar un vector de características con los vectores de su primera y segunda derivada mejoran sustancialmente la tasa de reconocimiento de un sistema de búsqueda de este tipo, donde la distancia puede caer por debajo de un umbral debido al parecido

parcial de los sonidos de ciertas palabras.

- Las técnicas de alineamiento que fueron utilizadas en este trabajo permiten llevar a cabo un alineamiento instantáneo y flexible. Instantáneo porque no acumula la distancia entre las secuencias de vectores, únicamente se tienen que monitorear el ultimo valor de la columna. También es flexible porque la alineación de cada marco de una matriz de referencia contra las matriz de la consulta, se lleva a cabo llenando únicamente una columna conforme se avanza en el tiempo, sin tener que llenar una matriz completa.

5.2. Trabajos Futuros

Con la experiencia obtenida e impulsada por la incursión tan basta de los investigadores en este problema, surgen algunas ideas que pensamos se pueden incorporar para el desarrollo de esquemas más robustos.

- Dado que el sistema presentado es dependiente del hablante, pensamos en la implementación de esquemas adicionales de extracción de características que puedan generar vectores de características más robustos y permitan que el sistema sea independiente del hablante.

- El hecho de tener que fijar un valor de cercanía antes de poder llevar a cabo una búsqueda, puede complicar el uso de esta herramienta. Por tanto se puede hacer uso de la cuantización vectorial VQ de los vectores de características para generar secuencias de números, donde estas secuencias pueden ser tratadas directamente como si fueran cadenas de texto, tal y como lo presentaron Rabiner y Juang en [Rabiner y Juang, 1993], donde aplican VQ a las secuencias de vectores para que un HMM discreto pueda ser entrenado y poder realizar la tarea de reconocimiento. Por tanto la aplicación de VQ a las matrices de características obtenidas en la fase de extracción de características de nuestro esquema de búsqueda, eliminaría el uso de dicho valor de cercanía ya que se puede comparar directamente si un número es igual a otro o no. Para poder llevar a cabo esto, es necesario contar con un algoritmo que se

encargue de particionar el espacio de vectores de características. El uso de técnicas de agrupamiento se vuelve medular en este esquema. Existen diferentes algoritmos probados en el reconocimiento de voz como el k-medias, donde uno tienen que definir el tamaño del libro de códigos CB (del Inglés *code book*). Existen otros algoritmos basados en gráfos, donde no se tiene que establecer el tamaño de dicho CB sino que el algoritmo crea grupos usando una medida de distancia, donde los elementos más cercanos conformarán un grupo.

- Otro esquema que se podría aplicar a este problema y que ha demostrado ser el mejor esquema para realizar reconocimiento de voz, son los Modelos Ocultos de Markov HMM. En un esquema de búsqueda de elocuciones de palabras o frases con HMM, podríamos ver a una consulta representada por su HMM y a los archivos de audio de referencia como secuencias de símbolos, producto de la VQ, que traduce vectores de características a números enteros que corresponden con los índices de cada centroide del libro de códigos. La búsqueda se podría llevar a cabo usando secciones de las secuencias de números (traducción de una matriz de características) para obtener la probabilidad de que el modelo de la consulta produzca dicha sección de la referencia.

Referencias

[Brown et al., 1996] Brown, M. G., Foote, J. T., Jones, G. J. F., Jones, K. S., y Young, S. J. (1996). Open-vocabulary speech indexing for voice and video mail retrieval. In *Proceedings of the Fourth ACM International Conference on Multimedia*, MULTIMEDIA '96, pages 307–316, New York, NY, USA. ACM.

[Burkhard y Keller, 1973] Burkhard, W. A. y Keller, R. M. (1973). Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236.

[Camarena, 2011] Camarena, J. A. (2011). Notas de síntesis y reconocimiento de voz.

[Camarena-Ibarrola y Chávez, 2006] Camarena-Ibarrola, A. y Chávez, E. (2006). On musical performances identification, entropy and string matching. In Gelbukh, A. y Reyes-Garcia, C., editors, *MICAI 2006: Advances in Artificial Intelligence*, volume 4293 of *Lecture Notes in Computer Science*, pages 952–962. Springer Berlin Heidelberg.

[Camarena-Ibarrola et al., 2009] Camarena-Ibarrola, A., Chávez, E., y Tellez, E. (2009). Robust radio broadcast monitoring using a multi-band spectral entropy signature. In Bayro-Corrochano, E. y Eklundh, J.-O., editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 5856 of *Lecture Notes in Computer Science*, pages 587–594. Springer Berlin Heidelberg.

[Can y Saraclar, 2011] Can, D. y Saraclar, M. (2011). Lattice indexing for spoken term detection. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(8):2338–2347.

[Chan y Lee, 2010] Chan, C.-a. y Lee, L.-s. (2010). Unsupervised spoken-term detection

with spoken queries using segment-based dynamic time warping. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Chan y Lee, 2011] Chan, C.-A. y Lee, L.-S. (2011). Integrating frame-based and segment-based dynamic time warping for unsupervised spoken term detection with spoken queries. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5652–5655.

[Chen et al., 2015] Chen, G., Parada, C., y Sainath, T. (2015). Query-by-example keyword spotting using long short-term memory networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5236–5240.

[Choi et al., 1998] Choi, J., Hindle, D., Hirschberg, J., Magrin-Chagnolleau, I., Nakatani, C. H., Pereira, F. C., Singhal, A., y Whittaker, S. (1998). Scan-speech content based audio navigator: a system overview. In *ICSLP*.

[Dmitry y Bovbel, 2007] Dmitry, M. y Bovbel, E. (2007). Indexing and retrieval scheme for content-based multimedia applications. In Matoušek, V. y Mautner, P., editors, *Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 162–169. Springer Berlin Heidelberg.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to {ROC} analysis. *Pattern Recognition Letters*, 27(8):861 – 874. {ROC} Analysis in Pattern Recognition.

[Fiscus et al., 2007] Fiscus, J. G., Ajot, J., Garofolo, J. S., y Doddingtion, G. (2007). Results of the 2006 spoken term detection evaluation. In *Proc. SIGIR*, volume 7, pages 51–57.

[Furui, 1986] Furui, S. (1986). Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 1991–1994.

[Garofolo et al., 2000] Garofolo, J. S., Auzanne, C. G., y Voorhees, E. M. (2000). The trec spoken document retrieval track: A success story. *NIST SPECIAL PUBLICATION SP*, 500(246):107–130.

[Glavitsch y Schäuble, 1992] Glavitsch, U. y Schäuble, P. (1992). A system for retrieving speech documents. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 168–176, New York, NY, USA. ACM.

[Guo y Siegelmann, 2004] Guo, A. y Siegelmann, H. (2004). Time-warped longest common subsequence algorithm for music retrieval.

[Hazen et al., 2009] Hazen, T., Shen, W., y White, C. (2009). Query-by-example spoken term detection using phonetic posteriorgram templates. In *Automatic Speech Recognition Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 421–426.

[Hermansky y Sharma, 1998] Hermansky, H. y Sharma, S. (1998). Traps – classifiers of temporal patterns. In *IN PROCEEDINGS OF 5TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING, ICSLP 98*, pages 1003–1006.

[Huang et al., 2001] Huang, X., Acero, A., y Hon, H.-W. (2001). *Spoken Language Processing a Guide to Theory, Algorithm, and System Development*. Prentice-Hall, 1 edition.

[Li y Liao, 2012] Li, W. y Liao, Q. (2012). Keyword-specific normalization based keyword spotting for spontaneous speech. In *ISCSLP*, pages 233–237.

[Liu et al., 2002] Liu, M., Wan, C., y Wang, L. (2002). Content-based audio classification and retrieval using a fuzzy logic system: towards multimedia search engines. *Soft Computing*, 6(5):357–364.

[Liu et al., 2004] Liu, M., Wan, C., y Wang, L. (2004). A fuzzy logic approach for content-based audio classification and boolean retrieval. In *Fuzzy Logic and the Internet*, pages 135–156. Springer.

[Mamou et al., 2007] Mamou, J., Ramabhadran, B., y Siohan, O. (2007). Vocabulary independent spoken term detection. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 615–622, New York, NY, USA. ACM.

[Mantena et al., 2014] Mantena, G., Achanta, S., y Prahallad, K. (2014). Query-by-example spoken term detection using frequency domain linear prediction and non-segmental dynamic time warping. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(5):946–955.

[Monroy et al., 2008] Monroy, L. R. M., Ledo, D. R. I. L. M., Belchez, D. G. H., Licona, L. E. S., Kan, D. F. L., Eslava, L. F. S., y Herrera, L. S. R. B. (2008). Cuaderno de apoyo. reforma constitucional en materia de justicia penal y seguiridad pública.

[Montejano y Cordero, 2008] Montejano, M. C. G. y Cordero, L. A. A. (2008). Estudio teórico-conceptual, de las principales iniciativas presentadas en la materia, de derecho comparado y de la reforma del estado.

[Moyal et al., 2013] Moyal, A., Aharonson, V., Tetariy, E., y Gishri, M. (2013). Keyword spotting methods. In *Phonetic Search Methods for Large Speech Databases*, SpringerBriefs in Electrical and Computer Engineering, pages 7–11. Springer New York.

[Navarro y Raffinot, 2002] Navarro, G. y Raffinot, M. (2002). *Flexible Pattern Matching in Strings – Practical on-line search algorithms for texts and biological sequences*. Cambridge University Press. ISBN 0-521-81307-7. 280 pages.

[Olsson y Oard, 2009] Olsson, J. S. y Oard, D. W. (2009). Combining lvcsr and vocabulary-independent ranked utterance retrieval for robust speech search. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 91–98, New York, NY, USA. ACM.

[Oppenheim et al., 1968] Oppenheim, A., Schafer, R., y Stockham, T.G., J. (1968). Nonlinear filtering of multiplied and convolved signals. *Proceedings of the IEEE*, 56(8):1264–1291.

[Parada et al., 2009] Parada, C., Sethy, A., y Ramabhadran, B. (2009). Query-by-example spoken term detection for oov terms. In *Automatic Speech Recognition Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 404–409.

[Rabiner, 1989] Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rabiner y Juang, 1993] Rabiner, L. y Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, 1 edition.

[Rabiner y Schafer, 1978] Rabiner, L. y Schafer, R. (1978). *Digital Processing of Speech Signals*. Prentice-Hall, 1 edition.

[Rivera, 2011] Rivera, R. C. (2011). Apuntes sobre el sistema penal acusatorio: Los juicios orales.

[Sakoe y Chiba, 1978] Sakoe, H. y Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49.

[Schafer, 2008] Schafer, R. (2008). Homomorphic systems and cepstrum analysis of speech. In Benesty, J., Sondhi, M., y Huang, Y., editors, *Springer Handbook of Speech Processing*, pages 161–180. Springer Berlin Heidelberg.

[Schäuble y Wechsler, 1995] Schäuble, P. y Wechsler, M. (1995). First experiences with a system for content based retrieval of information from speech recordings. In *Speech Recordings,? IJCAI-95 Workshop on Intelligent Multimedia Information Retrieval, M.Maybury (chair), working notes*, pages 59–69.

[Shen et al., 2009] Shen, W., White, C. M., y Hazen, T. J. (2009). A comparison of query-by-example methods for spoken term detection.

[Sigurdsson et al., 2006] Sigurdsson, S., Petersen, K. B., y Lehn-Schiøler, T. (2006). Mel frequency cepstral coefficients: An evaluation of robustness of mp3 encoded music. In *IN PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON MUSIC INFORMATION RETRIEVAL*.

[Silaghi, 2005] Silaghi, M.-C. (2005). Spotting subsequences matching an hmm using the average observation probability criteria with application to keyword spotting. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1118. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Singh et al., 2014] Singh, P., Manjunath, K., Ravi Kiran, R., Yadav, J., y Sreenivasa Rao, K. (2014). Indexing and retrieval of speech documents. In Kumar Kundu, M., Mohapatra, D. P., Konar, A., y Chakraborty, A., editors, *Advanced Computing, Networking and Informatics- Volume 1*, volume 27 of *Smart Innovation, Systems and Technologies*, pages 17–24. Springer International Publishing.

[Stevens et al., 1937] Stevens, S. S., Volkmann, J., y Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

[Terrazas, 2010] Terrazas, J. R. B. (2010). La reforma integral al sistema de justicia penal en el estado de chihuahua.

[Wechsler et al., 2000] Wechsler, M., Munteanu, E., y Schäuble, P. (2000). New approaches to spoken document retrieval. *Information Retrieval*, 3(3):173–188.

[William H. Press, 1992] William H. Press, Saul A. Teukolsky, W. T. V. B. P. F. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition.

[Yang, 2002] Yang, C. (2002). Efficient acoustic index for music retrieval with various degrees of similarity. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 584–591. ACM.

[Young et al., 2006] Young, S., Gales, G. E. M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., , Odell, J., Ollason, D., Povey, D., Valtchev, V., y Woodland, P. (2006). *The HTK Book*.

[Zhu et al., 2007] Zhu, Y., Ming, Z., y Huang, Q. (2007). Svm-based audio classification for content-based multimedia retrieval. In *Multimedia Content Analysis and Mining*, pages 474–482. Springer.